International Journal of Pattern Recognition and Artificial Intelligence

Analyzing Data Changes using Mean Shift Clustering

Nir Sharet
Department of Computer Science, University of Haifa, Haifa 31905, Israel

Ilan Shimshoni
Department of information Systems, University of Haifa, Haifa 31905, Israel
ishimshoni@mis.haifa.ac.il

A non-parametric unsupervised method for analyzing changes in complex datasets is proposed. It is based on the mean shift clustering algorithm. Mean shift is used to cluster the old and new datasets and compare the results in a non-parametric manner. Each point from the new dataset naturally belongs to a cluster of points from its dataset. The method is also able to find to which cluster the point belongs in the old dataset and use this information to report qualitative differences between that dataset and the new one. Changes in local cluster distribution are also reported. The report can then be used to try to understand the underlying reasons which caused the changes in the distributions. On the basis of this method, a transductive transfer learning method for automatically labeling data from the new dataset is also proposed. This labeled data is used, in addition to the old training set, to train a classifier better suited to the new dataset. The algorithm has been implemented and tested on simulated and real (a stereo image pair) datasets. Its performance was also compared to several state-of-the-art methods.

Keywords: Change detection; cluster analysis; mean shift clustering.

1. Introduction

Consider a general process which among other things produces datasets. The process could be, for example, a production and inspection procedure or a survey conducted on consumer buying habits. This data can then be analyzed and, if some of the examples are labeled, classifiers can be trained on them. Under the naïve assumption that the process does not change over time, the new datasets produced by the process will be drawn from the same distribution, the analysis performed on the initial dataset will remain valid, and the classifiers will maintain their performance.

The problem, however, is that processes vary over time and the distribution of the data may change. The change might be an indication that the underlying production process is not working correctly, requiring manual intervention by the operator to fix the problem. In order for the domain expert to be able to diagnose the problem, it is not enough to state that the distribution has changed. It is important to give a qualitative description of the nature of those changes. When a classifier has been trained on the initial dataset, changes in the distribution will diminish

its performance on the new dataset. Estimating the decrease in performance is important for deciding whether a new classifier has to be trained and whether that can be done automatically without requiring a large number of new manually labeled data points.

To address this problem we propose the following approach. A clustering algorithm is run on the dataset. The parameters of the algorithm are automatically set so that the results of the clustering will not change when the algorithm is run on different datasets drawn from the same distribution. It is possible that more than one set of parameters can be used, each representing a different aspect of the data by a different number of clusters.

For a given set of parameters two datasets are compared. Our technique compares the clusters in a non-parametric manner. It can detect whether a cluster has been split into several clusters or, conversely, whether several clusters merged into one. Newly created or eliminated clusters can also be detected, as can those that have moved or whose dispersion has changed. All these results are reported to the user, who can use them to try to understand the cause of the change in the distribution. The expert can then decide which if any corrective action should be taken to return the process to an adequate state. In some cases classifiers trained on the initial dataset can be retrained automatically for the new dataset without human intervention.

The basis of our algorithm is the mean shift clustering algorithm (MSC) [9,14]. This algorithm is used for several reasons. First, it is closely related to the non-parametric estimation of the distribution (kernel density estimation). Second, as the parameter of the algorithm (bandwidth) changes continuously, so does the cluster structure. This is not true, for example, for other clustering algorithms such as k-means or Gaussian mixture models. We propose a natural extension to MSC that compares two datasets in a non-parametric fashion.

The main contributions of the paper are as follows. The algorithm operates on large complex high dimensional datasets. Each dataset can be composed of several clusters of arbitrary shape. The algorithm is able to analyze the changes the clusters underwent from one dataset to another. Contrary to other algorithms it does not produce a single number that measures the global change between the datasets, rather it gives a description of the local changes that occurred between the datasets. This description can then be used by a higher level process or a domain expert to try to explain the reasons for the changes between the datasets.

The paper continues as follows. In the next section related work will be reviewed. A short description of the MSC algorithm and its important characteristics will also be given. Section 3 will describe our method for analyzing a single dataset, as a function of the bandwidth, yielding a tree structure of clusters. One of its main results is a set of bandwidth values which produce stable clustering results on datasets drawn from the same distribution. In Section 4 the more general technique for comparing two unlabeled datasets will be presented. The method can also be used in the transductive transfer learning setting. Initial results showing how our

method can be used in that setting are given in Section 5. There we are able to use the results of our algorithm to automatically train a classifier for the new dataset using the one trained for the old dataset. Experimental results on simulated and real data will be presented in Section 6. We use a stereo image pair as an example of two datasets that underwent some change and will apply the algorithm to them. In this section we will also compare our method to several other methods for comparing clusterings and distributions. Finally, in Section 7 we will draw some conclusions from our work and suggest future research directions.

2. Related work

When considering data change analysis, techniques can be divided into parametric and non-parametric tests. Parametric methods can be considered when the distributions from which the datasets were drawn can be approximated as belonging to a specific distribution. In this case there exist statistical methods which compare two datasets generated at different times. These methods can determine for example whether there has been a change in the mean or the variance of the distribution. When the distribution is complex applying these methods on the whole dataset the results might not be meaningful. They can be applied however on each cluster separately. We however, concentrate on non-parametric methods. We divide these methods into types based on the information known when the test is performed.

The first setting that should be considered is the supervised setting in which both datasets are labeled. In this paper we will not be considering this setting.

In the semi-supervised setting the initial dataset is labeled but we are not familiar with the new dataset. In this case we try to determine whether our knowledge is useful for the new dataset. In Section 5 we give an example on how our algorithm can be used in the semi-supervised setting but this is not the main focus of the paper.

A more relevant setting which is relevant for our case is when changes and differences between the datasets may be considered for discovering truly unexpected phenomena such as outliers. There are several research directions in outlier detection, as it was defined in [36]. *Distribution-based* techniques consider outliers as points which have low distribution density values. These techniques use a parametric distribution model of the old dataset estimated from it [3,32]. *Density-based* methods estimate the density in the old dataset using non-parametric methods such the Parzen window method [6]. *Distance-based* methods test whether there exist a certain number of points in the old dataset within a certain distance from a point in the new dataset. Points for which this condition does not hold are considered outliers [21,1]. Finally, the *clustering-based* approach applies a clustering algorithm to the data and considers clusters with significantly fewer points than other clusters as clusters of outliers [23,11]. Clustering data with outliers is in itself a challenging task. In [29] for example, a robust neural gas algorithm is proposed. It is able to deal with data with outliers. It then automatically decides which points are outliers using a

minimum description length (MDL) measure. Our work also performs clustering as a first step and analyzes the differences between the datasets using a method based on mean shift, which can be described as a density-based method. We do not, however, only search for outliers but also look for other types of changes in the distribution, for which the algorithm produces a qualitative description.

In the unsupervised setting only the two datasets are given, and an important research direction is to compare clustering results on the same dataset. This direction is related to our work: in the first part of the algorithm we find for which parameter values the clustering results have changed and then choose values in between them, for which we expect the clustering results to be stable when the algorithm is given different datasets generated from the same distribution. In our analysis we are able to give a qualitative description of the changes in the clustering results between the two datasets, for each of these values. This is in contrast to a large number of works which only give a numeric value for the difference between the clusterings. In [31,17,5,16] measures based on pair counting were suggested. Set matching based measures are proposed in [22] and in [25,26]. Information theoretic based measures were proposed in [35]. In [30] a spatially aware clustering comparison method was proposed. All these types of measures are intended for comparing results of clustering algorithms applied on the same data base. Such methods cannot be used in our case when an algorithm is applied to two different datasets. This constraint is lifted in the methods proposed in [2] and [8] but the algorithm still returns a numerical value and not a description of the changes.

A different approach is taken by measuring the Kullback-Leibler divergence between the two datasets. The method measures the difference between the distributions from which the two datasets have been drawn. Thus, any change between the distributions is measured. As one of the main goals of our algorithm is to determine whether a classifier trained on the first dataset can be used on the second, we are interested in the structure of the clusterings and the shape and positions of each cluster, but not, for example, in the number of points in each cluster — which the KL divergence also measures. As in all the other methods, this one also returns a single value and not a description of the difference between the datasets.

It is also possible to use classifiers such as a one class SVM to define all the points of the initial dataset as belonging to one class. Change is detected if the new points are classified as not belonging to that class. In [24], several one class SVM classifiers are trained on different representations of the initial dataset. For new data points (e.g., in the time-series setting), if all the classifiers classify the point as not belonging to the class, it is declared an outlier or a novelty point.

An important field of research deals with change detection in data streams [20,28,4,33,13,12]. Even though they are also interested in change detection, the setting and emphasis is different. In stream analysis at each point in time a new data point is generated while in our setting all the points of a dataset are received at once. One of their main goals is to detect change as fast as possible

(i.e., with the smallest number of points), where in our case the two point sets are given. Our main goal is to analyze complex multi-cluster distributions and produce a description of the change. Thus, even though techniques might be shared between the methods designed to solve the two problems, the goals and the emphases are different.

2.1. *Mean shift clustering*

There exist many clustering algorithms, such as k-means clustering, spectral clustering, agglomerative clustering, and more, each with its strengths and weaknesses. We chose to use mean shift clustering as the basis of our algorithm. Mean shift has been used in hundreds of applications, among them mean shift segmentation [7] and mean shift tracking [10]. We will therefore give a short description of this algorithm and its main characteristics. For a complete description of the algorithm please refer to [9].

Given n d-dimensional data points $\mathbf{x}_i, i = 1, \ldots, n$, the kernel density estimator method approximates the density f at point $\mathbf{x}$ as

$$f(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} k\left(\frac{||\mathbf{x} - \mathbf{x}_i||^2}{h} \right), \tag{1}$$

where $k()$ is the profile of a kernel function $K()$. A d-variate kernel is a bounded function with compact support satisfying

$$K(\mathbf{x}) \geq 0 \text{ and } \int_{\Re^d} K(\mathbf{x}) d\mathbf{x} = 1.$$

h is the bandwidth value. The bandwidth parameter h acts as a smoothing parameter. The larger the value, the smoother the estimated density function. The mean shift clustering method applies a simple gradient ascent of the density function starting from every data point. The gradient is computed by taking the derivative of eq.1. The process is applied to all the points in the dataset. For each data point in the dataset it starts with $\mathbf{y}_1$ equals to that point. At each iteration $\mathbf{y}_{j+1}$ is computed as follows:

$$\mathbf{y}_{j+1} = \frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\frac{||\mathbf{y}_j - \mathbf{x}_i||}{h} \right)}{\sum_{i=1}^{n} g\left(\frac{||\mathbf{y}_j - \mathbf{x}_i||}{h} \right)} \quad j = 1, 2, \ldots , \tag{2}$$

where $g(x) = -k'(x)$. The process converges to a local maximum of the density function (a mode). All points which converge to the same mode are declared as belonging to the same cluster. Points which converge to very small clusters are defined as not belonging to any cluster. These points will belong to a special cluster called Cluster_0. The user selects the value of the bandwidth parameter h. This value indirectly determines the number of clusters. As h increases in value, the density function estimate becomes smoother, which results in fewer clusters.

Under the assumption that the kernel function K has a limited support, each iteration of the mean shift algorithm can be computed efficiently using approximate nearest neighbor techniques [14].

The main characteristics of the mean shift algorithm are:

(1) The algorithm is closely related to the kernel density estimation function. For each value of the bandwidth h, clustering is performed, which finds for each mode the points belonging to its basin of attraction.
(2) The number of clusters is not given by the user; rather, it is determined by the algorithm as a function of h.
(3) Continuous changes in the value of h yield continuous changes in the clustering results.

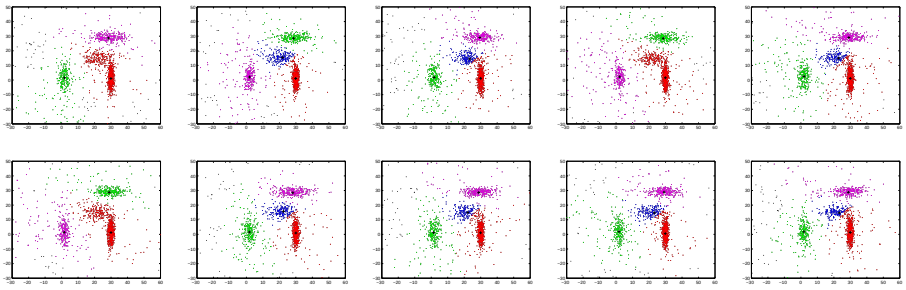
These important characteristics will be used by our change detection algorithm.

3. Single dataset analysis

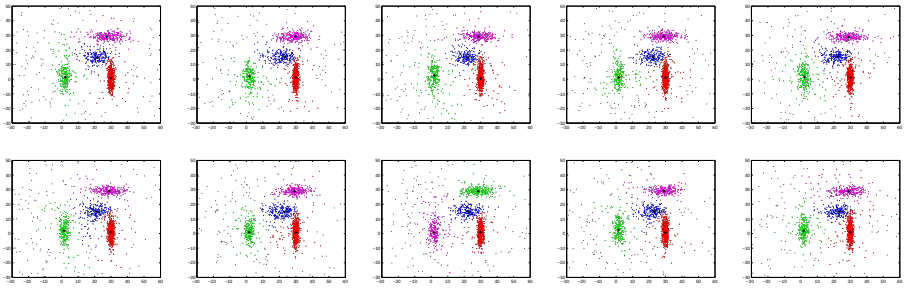
In this section we will describe our method for the analysis of a single unlabeled dataset. As mentioned above, the outcome of MSC is influenced by the bandwidth h, which determines the smoothness of the calculated density function. As h changes, so does the underlying density function. It is therefore important to develop a method for selecting the appropriate bandwidth or bandwidths. For a given dataset we can use several h's that represent different aspects of the structure of the data. The bandwidth value should be within a range where the cluster structure is stable. I.e., for different datasets drawn from the same distribution, the algorithm should return the same number of clusters in approximately the same places in space. We want to avoid using bandwidth values where phase changes occur. In other words, we do not want to use an h value where around that value the cluster structure of the data changes. At these values two datasets drawn from the same distribution might yield different cluster structures, causing us to mistakenly detect a change in the distribution. We denote this bandwidth value set as CP (change points).

In order to demonstrate the importance of choosing stable h values, consider the dataset shown in Figure 1. For this dataset, for values of h greater than 8.9 three clusters are generated. Around the bandwidth value $h = 8.9$ one of the three recovered clusters splits into two. At $h = 2.9$ one of the four clusters splits again into two clusters. We generated 10 datasets drawn from the same distribution and ran MSC on them using the unstable bandwidth $h = 8.9$ and the stable bandwidth value in between the two unstable values, $h = (8.9 + 2.9)/2 = 5.9$. In the first case, for 7 datasets 3 clusters were found, whereas for the rest 4 clusters were found. In the second case, 4 clusters were found for all 10 datasets. Thus, the h values which lie at the midpoint between two distant change points can be selected as stable representatives to describe the dataset.

In order to find the appropriate bandwidth values, we run the $finding\_h$ algorithm shown in Algorithm 1, as follows. We run MSC for bandwidth values between



(a) Unstable bandwidth $h = 8.9$



(b) Stable bandwidth $h = 5.9$

Fig. 1. When given two datasets drawn from the same distribution for bandwidth value h close to the phase change ($h = 8.9$), clustering results can be different as seen in (a). In some cases three clusters are produced while in others four. For a stable value of h ($h = 5.9$), mean shift consistently produces the same number of clusters (b).

finding_h($\mathbf{h_{max}}, \mathbf{h_{min}}, \mathbf{\Delta}, \mathbf{Data}$)

$res[h] \leftarrow MSC(h_{max}, Data)$
$CP \leftarrow h_{max}$
for $h \leftarrow h_{max}$ **to** h_{min} **do**
  $h \leftarrow h - \Delta$
  $res[h] \leftarrow MSC(h, Data)$
  if $structure\_changed(res[h + \Delta], res[h])$ **then**
    $CP \leftarrow CP \bigcup \{h\}$;
  end
end

Algorithm 1: finding_h($\mathbf{h_{max}}, \mathbf{h_{min}}, \mathbf{\Delta}, \mathbf{Data}$)

h_{max} and h_{min} at a resolution of Δ_h. Each consecutive pair of results is compared by putting the results in a confusion matrix. Each point is placed in the cell in the matrix representing the cluster it belonged to in the first run and the cluster it belonged to in the second run. We use a membership test to match clusters from

the first run to clusters of the second run. If, for example, most of the points belonging to cluster A in the first run belong to cluster B in the second run, the clusters match. In addition, we are able to detect cases in which a cluster from the first run splits into several clusters in the second run or disintegrates into a set of points that do not cluster (denoting such points as belonging to cluster_0). A cluster tree is created as a result. The clusters are also named in order to represent the tree structure. The clusters generated for $h = h_{max}$ are named 1,2, etc. When, for example, cluster 1 splits into two clusters, they will be named 1.1 and 1.2. The h values at which the cluster structure change occurred are stored in a list of change points termed CP.

As mentioned above, stable points on which to perform comparisons to other datasets are chosen to be at midpoints between consecutive change points—as far as possible from the change points themselves. Such points are only selected if the distance between the change points is larger than $k\Delta_h$. In our implementation we used $k = 2$. The set of stable bandwidth values is denoted by SB. The stability of the clustering can also be verified using methods such as [18]. For unsupervised data all SB values can be used to analyze the data and represent the different aspects of its structure. When additional knowledge is available about the dataset —labeled data, for example—a meaningful subset of SB can be chosen. Details of this case will be discussed in Section 5.

The values of h_{max} and h_{min} are obviously data dependent. The value of h_{max} is selected such as the minimal (maybe only one) number of clusters is generated, while h_{min} has to be set so as the maximal number of clusters is generated. The resolution of the scan is set by Δ_h. If it is not set small enough several changes in clusters structure will be detected together.

Results of the analysis of a dataset are shown in Figures 2 & 3. Figure 2 shows how the clustering results change as a function of h. In this experiment $h_{max} = 40.0$, $h_{min} = 1.3$, and $\Delta_h = 0.3$ were used. For $h = 40$ only one cluster exists. At $h = 17.2$ the cluster splits into two. The splitting process continues for several other values of h. At $h = 6.7$ one of the clusters disintegrates. The results are also shown in a tree structure in Figure 3. The CP of the dataset is $CP = \{40, 17.2, 16.3, 15.4, 13.9, 13.6, 11.2, 10, 6.7, 2.5, 1.6\}$. The SB for this dataset is: $SB = \{40, 28.6, 16.75, 15.85, 14.65, 12.4, 10.6, 8.35, 4.6, 2.05\}$.

4. Data set comparison

The main challenge that we address in this paper is how to compare two datasets. These sets will be denoted D and D'. This problem is more challenging than the one we faced in the previous section where we analyzed the difference in the clustering results of a single dataset when the bandwidth value changed. Here we are given two different datasets consisting of different points and thus a different method must be devised to compare the different clustering results. There are two main differences between the two cases. First, the datasets are different and thus a simple set mem-

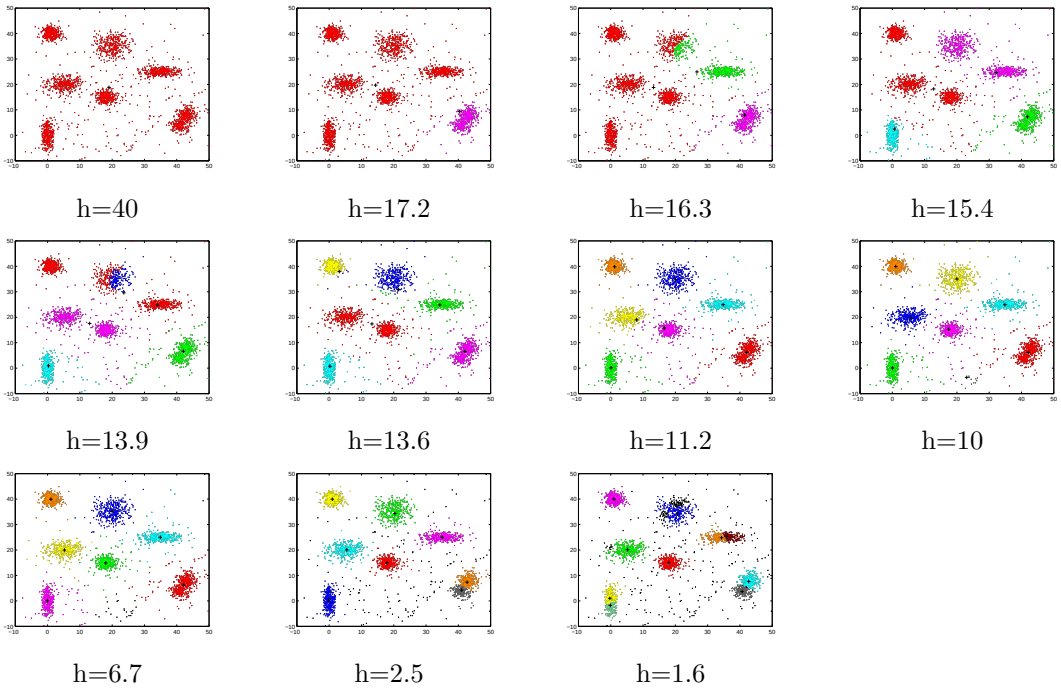


Fig. 2. MSC results with decreasing h values. The mode of each cluster is marked by a +.

bership test is not possible. Second, the cluster structure in this case can change in various ways, whereas in the previous case clusters could only split or disintegrate. Now they can split, merge, disintegrate or be formed. The distribution of points within a cluster may also change over time. Clusters can also move, contract or expand. Our goal is to detect all these changes without making any assumptions on the distribution of each cluster, in a non-parametric manner.

Here again we will use mean shift as the basis of our algorithm. The general idea is that each point $p' \in D'$ will be associated with a cluster of points from D and a cluster of points from D'. Analyzing these associations enables the algorithm to determine whether there has been a significant change between the datasets and give a qualitative description of the change.

4.1. *Two-set mean shift clustering*

The first step of the algorithm is to choose a set of h's which are stable for both datasets. This is done simply by running the single dataset process on both datasets, yielding the two change point lists, CP_D (Figures 2 and 3) and $CP_{D'}$ (Figures 4 and 5), which are then merged. From it the list of stable points, $SB_{D,D'}$, is computed.

For each $h \in SB_{D,D'}$ we apply the *two-set mean shift clustering* comparison

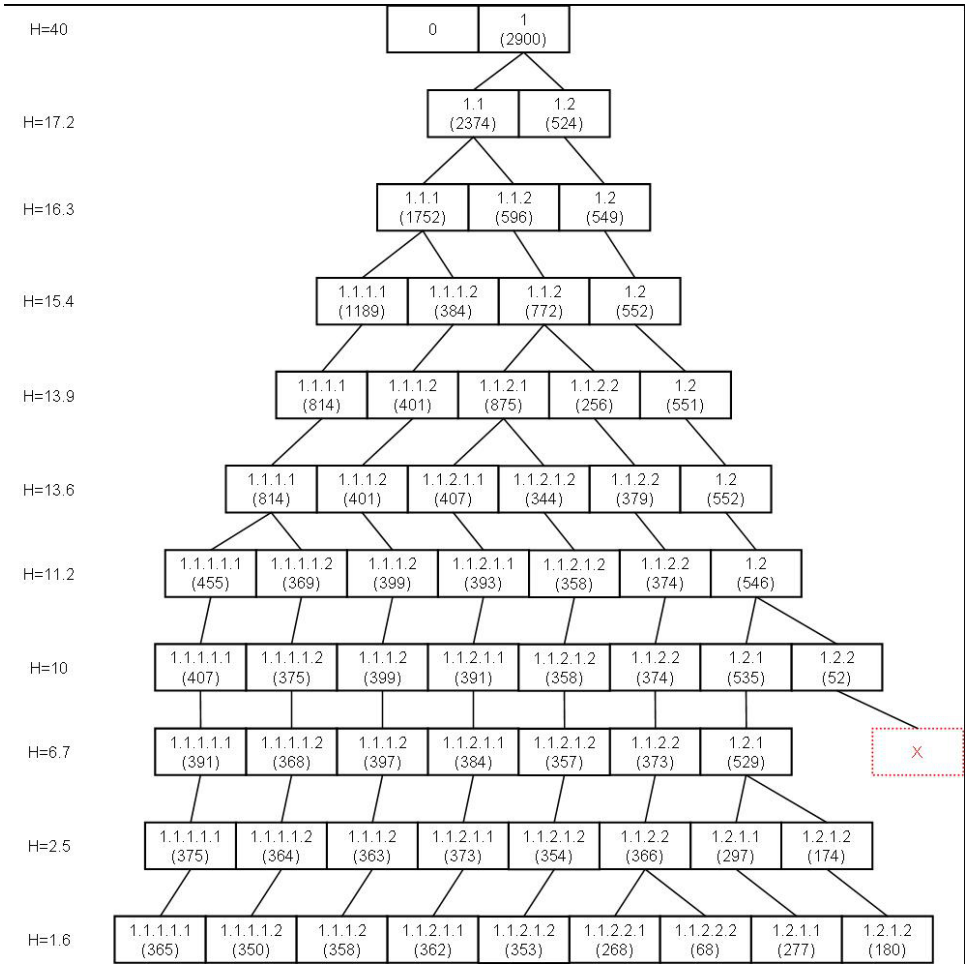


Fig. 3. The cluster structure tree of the dataset

procedure, $TSMSC(D, D', h)$ (shown in Algorithm 2), which will be described shortly. The basic function that we will be using is $MSC_p(p, h, D)$ that runs the mean shift procedure starting from point p, on dataset D, using bandwidth value h. The gradient ascent procedure is run until the process converges to a mode of the distribution of D and returns the ID of the cluster associated with that mode. In the standard mean shift procedure, it is assumed that $p \in D$. This assumption will be lifted in $TSMSC(D, D', h)$.

We will now describe the $TSMSC(D, D', h)$ procedure. At first, $MSC_p(p', h, D')$ is run starting from each point $p' \in D'$. In addition, $MSC_p(p', h, D)$ is also run, starting again from each point $p' \in D'$, seek-

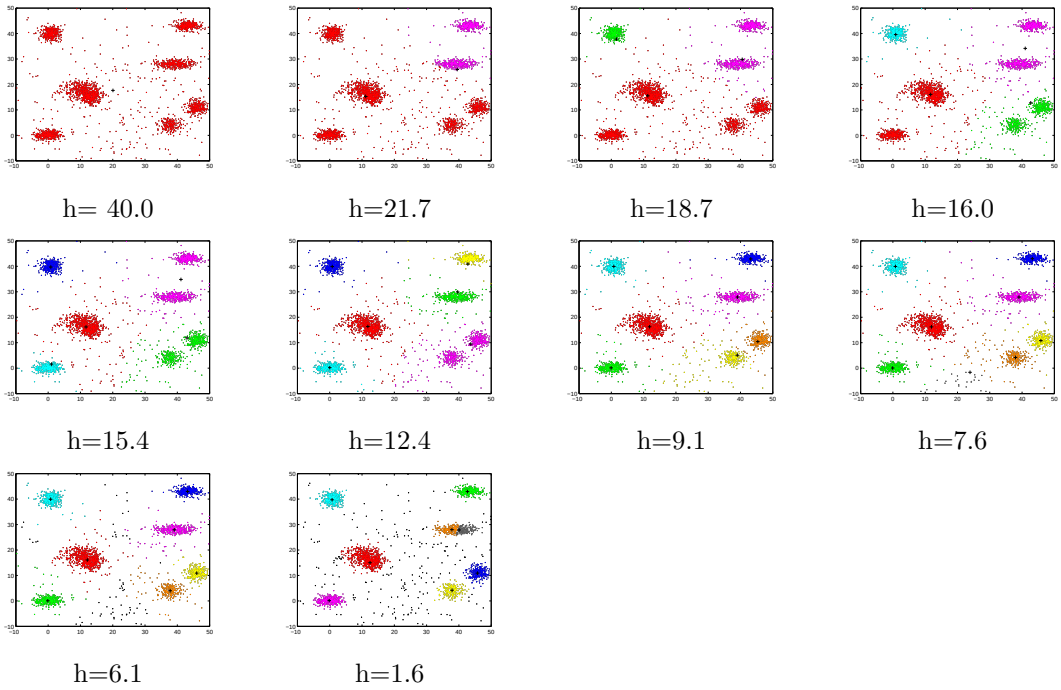


Fig. 4. MSC results with decreasing h values D'

TSMSC $(\mathbf{D}, \mathbf{D'}, \mathbf{h})$

foreach p' *in* D' **do**
  $C \leftarrow MSC_p(p', h, D)$
  $C' \leftarrow MSC_p(p', h, D')$
  $table(C', C) \leftarrow table(C', C) + 1$
end

Algorithm 2: TSMSC $(\mathbf{D}, \mathbf{D'}, \mathbf{h})$

ing modes in the D dataset (estimating the density using the points from dataset D). As a result, each point in D' is associated with two clusters: C', which is a cluster of D', and C, which is a cluster of D. Taking the results obtained for all the points, a confusion matrix (table) is generated and analyzed.

The running time of a $TSMSC(D, D', h)$ function is practically equivalent to running mean shift clustering twice. Mean shift is currently being used in many applications on large datasets. In a previous work we developed a mean shift algorithm which is able to deal with high-dimensional data [14] using approximate nearest neighbor data structures [19].

The analysis of the confusion matrix is performed as follows. Consider, for ex-

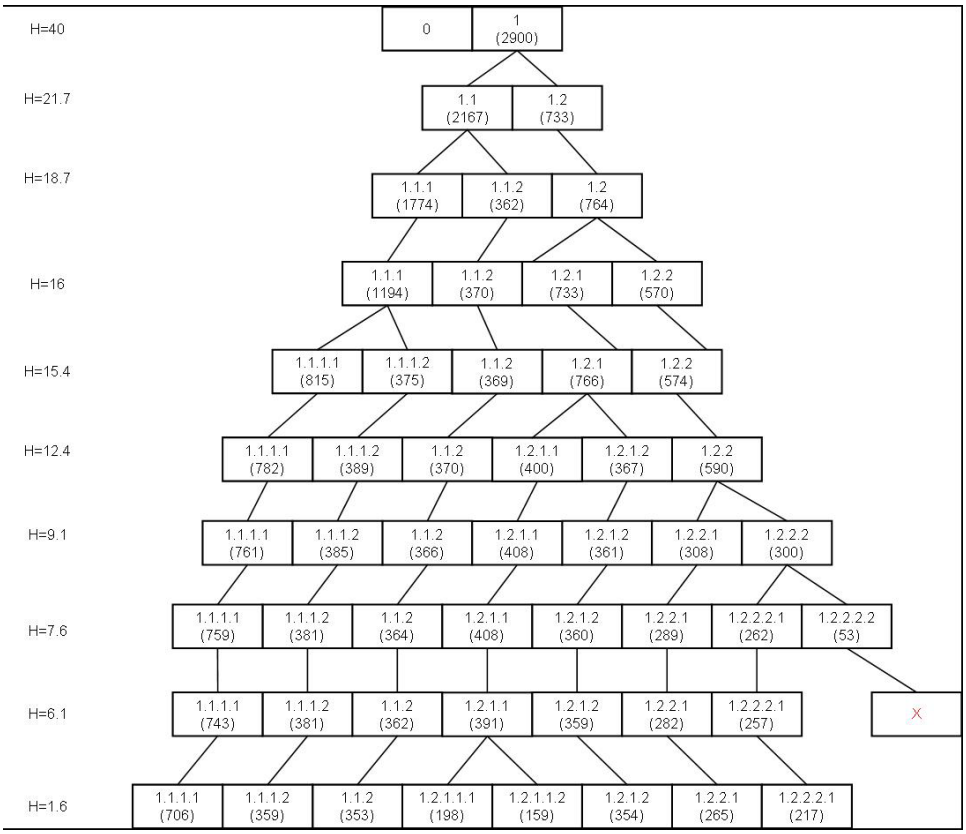


Fig. 5. The cluster structure tree with decreasing h values (D')

ample, that a new cluster has been formed in D' which did not exist in D (and which might later be discovered to be a cluster of outliers). Points belonging to this cluster converged to this new cluster when the regular mean shift process was run on D'. When, however, the process was run on those points on the distribution of D, the process did not converge to any mode (cluster_0) as there are no points in that region. Inspecting the confusion matrix we can conclude that a new cluster has been formed. In a similar fashion we can detect whether a cluster has disintegrated, whether a cluster has split into two clusters, or whether two clusters were merged to form a single cluster.

Clusters may also change their local distribution. In this case some of the points belonging to the cluster in D' will belong to cluster_0 in D. In addition, points which lie on the boundary of the cluster for one dataset may lie in the core of the cluster in the second. We therefore provide a non-parametric definition of a boundary point in the next section and use it in order to refine the analysis of this case. Thus, the confusion matrix we produce differentiates between points which lie

in the core of the cluster C' and points which lie on the boundary C'_b.

4.2. *Analyzing the cluster boundary*

When there are no significant changes between the datasets D and D', the clusters from D' will correspond to clusters from D. Nonetheless, despite this correspondence, some of the changes in the dispersion may not be detected. We want to be able to detect whether one of the cluster's positions or its dispersion has changed.

Changes in the dispersion can be discovered by detecting whether the boundary of the cluster has changed. To that end, we want to define which points lie on the boundary of the cluster. We will refer to those points as *boundary points*. The *boundary points* of cluster C will be denoted as C_b. We can analyze the changes in the boundary in two ways: by checking whether points from cluster C' in D' that converged to cluster C in D lie on C's boundary, and by discovering to where C's boundary points converge. By integrating the results of this analysis into the comparison process, we can detect changes in a cluster's dispersion and position.

Several methods have been suggested to determine when a point lies on a cluster's boundary. In general all these methods define a boundary point as a point with low probability density. In the parametric case there exists an estimated density function which can be used to compute the density at that point. For example, when considering the Gaussian distribution, the Mahalanobis distance can be used. In the non-parametric approach, methods like the *Tukey depth*, which is also known as *location* or *halfspace depth*, were suggested [34]. For a non-degenerate point set in space, the Tukey depth of a point p is the minimum number of data points on any side of a hyper-plane through p.

We can calculate the Tukey depth of each point in the cluster. Points with Tukey depth less than a threshold T will be considered to lie on the boundary. There are two main problems with this technique. First, in high dimensions $d \geq 3$, the expected time bound is $O(n^{d-1})$, which is computationally expensive. Second, the Tukey depth for a point is influenced also by the shape of the cluster. For example, points that lie in non-convex areas will not be considered to lie on the boundary, as the method considers all the data points and does not examine only the local area. Figure 6 illustrates the problem of using Tukey depth for a non-convex cluster. Even though both points lie on the boundary, only P_i will be classified as a boundary point by the Tukey depth, while P_k will not.

To find the identity of each point's cluster using MSC, we developed the h_{count} indicator to test whether the point is on the cluster's boundary. The h_{count} is defined as the number of points within radius h of the examined point that are associated with the same cluster. If h_{count} is smaller than a constant K, the point is considered to be on the boundary of its cluster. K can be set adaptively with respect to the cluster's characteristics by setting it as a certain percentage of the points belonging to the cluster. In addition, if within this ball there exists a point from another cluster, the point is also defined as a boundary point.

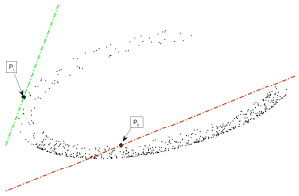
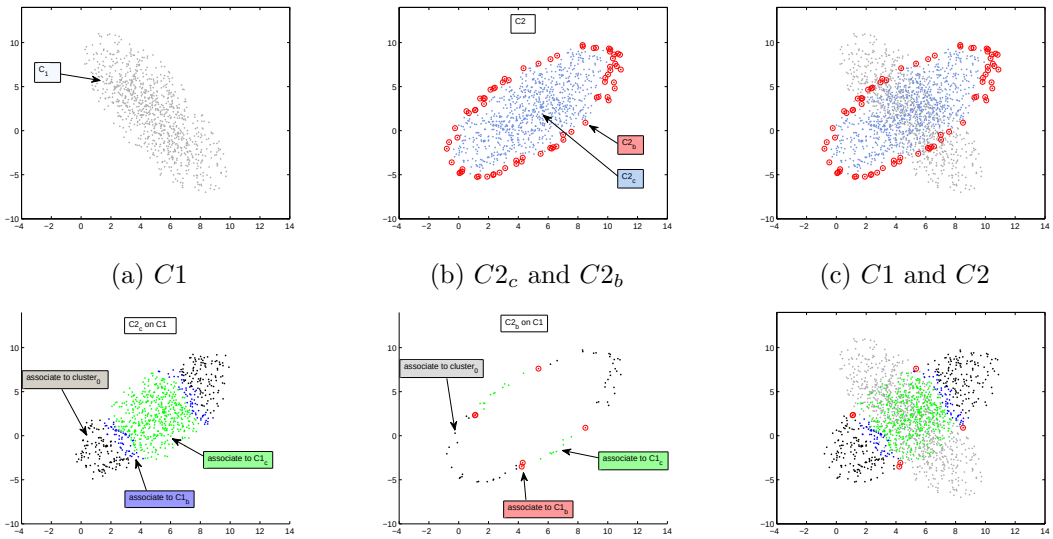


Fig. 6. *Tukey depth* on a non-convex cluster. Even though both points are boundary points, P_k will not be considered a boundary point by the Tukey depth method.



(a) $C1$ (b) $C2_c$ and $C2_b$ (c) $C1$ and $C2$

(d) convergence of points from $C2_c$ (e) convergence of points from $C2_b$ (f) convergence of points from $C1$, $C2_b$ and $C2_c$

Fig. 7. Comparison of two clusters with dispersion changes

Integrating this definition in the two dataset comparison process facilitates better analysis of the dispersion changes. Confusion matrices for two clusters with dispersion reduction or increase are shown in Table 1, where X indicates many points, $C1$ represents a cluster in D and $C2$ a cluster in D'. $C2_c$ indicates the points that belong to cluster $C2$ but do not lie on its boundary, while $C2_b$ indicates points that lie on cluster $C2$'s boundary. Analyzing the confusion matrix, we can deduce whether there has been a major change in the dispersion of the cluster and what change has occurred. Figure 7 illustrates a two cluster comparison with dispersion changes whose confusion matrix is given in Table 2. Looking at the result

we can see that many points from $C2$ and its boundary belong to cluster_0 in D indicating that the cluster's shape has changed. This is also evident from seeing that the boundaries of $C2$ and $C1$ do not coincide anymore.

Minor changes

	0	**C1**	**C1_b**
C2		X	
C2_b			X

(a) Most of the points from $C2_c$ converge to $C1_c$ and points from $C2_b$ converge to $C1_b$.

Dispersion reduction

	0	**C1**	**C1_b**
C2		X	
C2_b		X	

(b) Most of the points from $C2_c$ and $C2_b$ converge to $C1_c$.

Dispersion increase

	0	**C1**	**C1_b**
C2		X	X
C2_b	X		

(c) Most of the points from $C2_c$ converge to $C1_c$ and to $C1_b$. Points from $C2_b$ converge to cluster_0.

Table 1. Confusion matrices with dispersion changes

	0	**C1**	**C1_b**
C2	229	457	115
C2_b	47	19	6

Table 2. Output of the confusion matrix

4.3. *Computational complexity of the algorithm*

The algorithm is given as input two datasets D and D'. The algorithm is divided into two main components, the single dataset analysis component and the dataset comparison component. The first component consists of running the mean shift (MSC) algorithm several times, whereas the second component consists of running the two set mean shift (TSMSC) algorithm. The algorithm also performs analysis on confusion matrices but the complexity of this part is negligible compared to the other two. Since as we mentioned above the TSMSC algorithm is equivalent to running the MSC algorithm twice all that needs to be analyzed is the complexity of the MSC algorithm. For simplicity we will assume that $O(|D|) = O(|D'|)$.

The mean shift procedure is applied on all points in D. For each such point the iterative process given in eq. 2 is computed several times. In each iteration $\mathbf{y}_{j+1}$ is computed until convergence. The number of these iterations depends on dataset D but not on its size. Let It denote this number. Even though eq. 2 is evaluated on all the points of dataset D, only points $\mathbf{x}_i$ close to $\mathbf{y}_j$ contribute to the value of $\mathbf{y}_{j+1}$. This is because the weight $g(\frac{||\mathbf{y}_j - \mathbf{x}_i||}{h})$ decreases rapidly as $\frac{||\mathbf{y}_j - \mathbf{x}_i||}{h}$ increases. Moreover, for some of the most commonly used kernel functions $g()$ $g(a) \equiv 0$ for $a > 1$. Thus, in each iteration the algorithm first extracts from D a subset of the points close to $\mathbf{y}_j$ and then evaluates eq. 2 only on them. Thus, the complexity of the

naive implementation of the algorithm is $O(|D|It|D|d)$, where d is the dimension of the data and the complexity of the procedure which recovers the near neighbors of $\mathbf{y}_j$ is $O(|D|d)$. In our previous work [14] we developed a mean shift algorithm in which the near neighbors procedure was replaced by an approximate near neighbors algorithm. In our algorithm the Locality Sensitive Hashing (LSH) data structure was used [19,15]. The complexity of an LSH query is $O(d\log_2(|D|)/\epsilon^2)$, where ϵ is a measure of the error guaranteed by the LSH data structure. Thus, when LSH is used, the complexity of the MSC algorithm is $O(|D|It\,d\log_2(|D|)/\epsilon^2)$. Similar results can be obtained when other approximate nearest neighbors data structures are used. As our algorithm mainly consists of running MSC several times, this is also its computational complexity.

5. Dataset comparison in a transductive learning setting

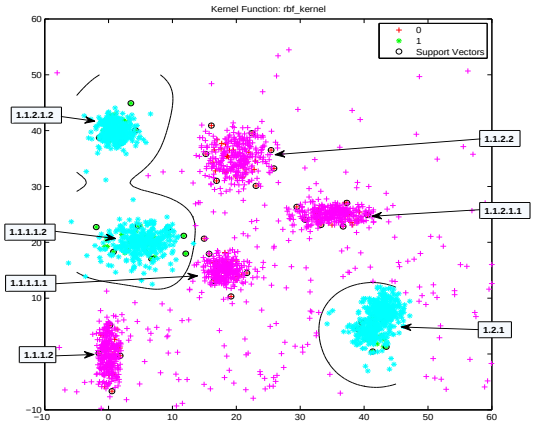
The proposed method can also be used in the transductive learning setting. A demonstration on how this can be done will now be given.

Training a classifier on the initial data set D using labeled data can facilitate its analysis. When the new dataset D' is obtained, a key question is whether the classifier trained on D can also be used on D' to obtain results of similar quality and, if not, whether the classifier could be retrained with no or minimal user intervention to produce a classifier better suited for the new dataset. The additional information we have in this case (the labeled points and the classifier) can help us perform the analysis described in Section 4, the results of which can be used to answer these questions.

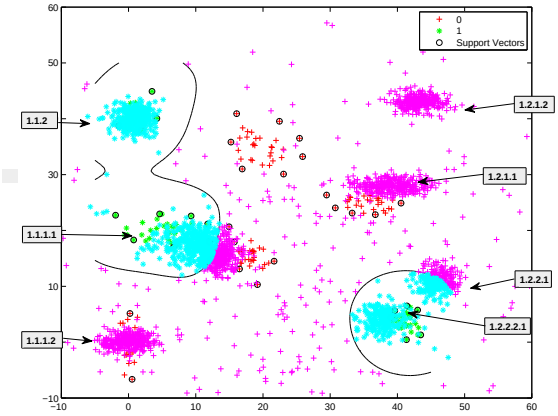
In the first step we can use this additional information in order to rank the set of stable bandwidths SB according to the correspondence between the cluster structure and the labels of points belonging to each cluster. The $TSMSC$ process will only be run on clustering results where nearly all the points belonging to each cluster have a single label. After the $TSMSC$ has been executed on the two datasets, we assume that points belonging to matching clusters should be classified with the same label. This conjecture is easily checked by applying the classifier to points in the corresponding clusters. Only if this condition does not hold is further investigation required.

Figure 8(a) shows the results of training an SVM classifier using the RBF kernel on the initial dataset D. Figure 8(b) shows the classification results obtained on D'. As described in Section 6.1, there have been quite a few changes in the underlying distribution between D and D'. Only a few of them, however, are related to the classifier.

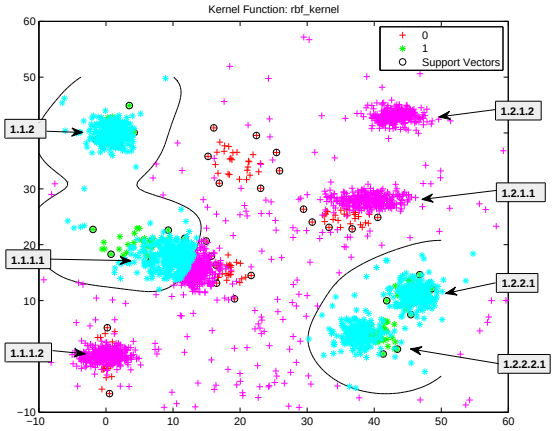
When a cluster (D1.2.1), which lies on one side of the class boundary, matches cluster(s) (D'1.2.2.1 & D'1.2.2.2.1) which lie on both sides of the class boundary, we will assume that the distribution of the cluster changed and a new classifier has to be trained. This can be done by automatically labeling points from the cluster from D' with the label of the corresponding cluster from D and adding them to the



(a) Original classifier results on D



(b) Original classifier results on D'



(c) Modified classifier results on D'

Fig. 8. Classifying old and new datasets: (a) Dataset D classified using training examples from D. (b) Dataset D' classified using the same classifier. (c) Classification results of D' using a modified classifier with additional training examples from D' which were automatically labeled.

training set of the classifier. This results in the classifier whose results are shown in Figure 8(c).

When two clusters from D ($D1.1.1.1.1$ & $D1.1.1.1.2$) with different labels are merged into a single cluster in D' ($D'1.1.1.1$), there is no point in retraining the classifier. Our assumption is that the change in data distribution made the classification problem harder. Some of the points of the new cluster now lie close to the class boundary and we should expect the classifier's error rate to increase as a result.

Finally, when a new cluster appears in D' ($D'1.2.1.2$), we can assume it does not constitute a problem if the vast majority of its points are classified as belonging to one class. However, it is perfectly possible that the points of this new cluster might instead be outliers. We would therefore recommend that the domain expert manually label a few points belonging to the new cluster before proceeding. Other changes in the distribution of cluster points of corresponding clusters (e.g., $D1.1.1.2$ & $D'1.1.1.2$) can be ignored if they are not related to the classification problem.

In conclusion, the results of the analysis can be used by the domain expert to minimize the number of examples which have to be manually labeled in order to maintain classification quality.

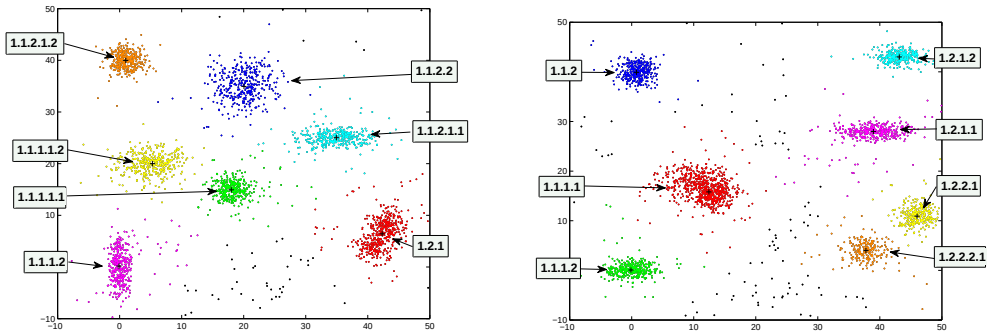
6. Comparisons and experimental results

In order to demonstrate the quality of our algorithm we will present in this section three types of results. First, we will apply our algorithm to two synthetic datasets to demonstrate its ability to detect several types of changes between them. In Section 6.2 we will compare our method to the methods referred to in Section 2 and show the strengths and weaknesses of our method. In Section 6.3 we will present the results of running our algorithm on real data (a stereo image pair) and show that the algorithm is able to recover the expected changes between the datasets without using any prior knowledge derived from the theory of stereo imaging.

6.1. *Results on synthetic data*

To demonstrate the results of our algorithm we will compare the two datasets D and D' shown in Figure 9. We begin by finding CP
$CP_D : \{40, 17.2, 16.3, 15.4, 13.9, 13.6, 11.2, 10, 6.7, 2.5, 1.6\}$ and
$CP_{D'} : \{40, 21.7, 18.7, 16, 15.4, 12.4, 9.1, 7.6, 6.1, 1.6\}$.
The common stable bandwidth values found are:
commonSB:{40,30.85,20.2,17.95,16.75,14.65,13,11.8,
10.6,9.55,8.35,7.15,4.3,2.05}.

We will now present the results of running $TSMSC$ for $h = 4.3$. The algorithm was run on all the points from dataset D'. The results are presented as a confusion matrix in Table 3. The columns show the results when the algorithm was run on distribution generated from dataset D (D clusters) and points belonging to their boundary, and the rows show the D' clusters. The confusion matrix is analyzed automatically according to the following rules. If most (more than 80% in our implementation) points from one cluster is matched to another cluster then the clusters are considered matched. If more than one cluster is matched to the same cluster then a merge (or a split) has occurred. If most of the points in a cluster are matched to cluster_0, the cluster disintegrated (or was formed). If a certain amount of points from a cluster (10% in our implementation) are matched to the boundary of its matched cluster or cluster_0 then we conclude that the underlying distribution of the cluster has changed between the datasets.



(a) D (b) D'

Fig. 9. Clustering results for the two datasets for $h = 4.3$

	0	1.1.2.1.2	1.1.2.1.2 B	1.1.2.2	1.1.2.2 B	1.1.2.1.1	1.1.2.1.1 B	1.1.1.1.2	1.1.1.1.2 B	1.1.1.1.1	1.1.1.1.1 B	1.2.1	1.2.1 B	1.1.1.2	1.1.1.2 B
0	164		4	7	10	4	9		7		6				3
1.1.2		346													
1.1.2-B		1	7												
1.2.1.1	2					308	50								
1.2.1.1-B	4					7									
1.1.1.1								203	108	175	230				
1.1.1.1-B	4			1	1			2	2	5	1				
1.2.2.2.1	2											192	18		
1.2.2.2.1-B	5											7	2		
1.2.2.1	2											239	23		
1.2.2.1-B	1						1					4	4		
1.1.1.2	2													352	7
1.1.1.2-B	3													6	2
1.2.1.2	349														
1.2.1.2-B	8														

Table 3. Confusion matrix for D and D' for $h = 4.3$. The columns represent D clusters while the rows represent D' clusters. Rows and columns marked with B represent points belonging the boundary of the cluster. The value K for the h_{count} boundary is the second percentile of each cluster.

Using these rules the automatic analysis off the confusion matrix from the perspective of D' (rows) yields:
Cluster D'1.1.1.1 (red) is the merger of clusters D1.1.1.1.1 and D1.1.1.1.2.
Cluster D'1.2.1.1 (pink) is mapped to D1.1.2.1.1 but some of the points are on the boundary, indicating that the cluster moved or changed its distribution.
Cluster D'1.1.1.2 (green) is mapped to D1.1.1.2.
Cluster D'1.2.1.2 (cyan) is a new cluster. All the points were associated with cluster_0.
Cluster D'1.1.2 (blue) is mapped to D1.1.2.1.2.
Cluster D'1.2.2.1 (yellow) is mapped to D1.2.1 but some of the points are on the boundary, indicating that the cluster moved or changed its distribution.
Cluster D'1.2.2.2.1 (orange) is mapped to D1.2.1 but some of the points are on the boundary, indicating that the cluster moved or changed its distribution.

Analyzing the matrix from the perspective of D (columns) yields:
Cluster $D1.2.1$ (red) is the merger of clusters $D'1.2.2.1$ and $D'1.2.2.2.1$.
Cluster $D1.1.2.2$ (blue) did not map to any cluster. This indicates that it disintegrated.

6.2. *Experimental comparison to other methods*

Most related works, such as the ones reviewed in Section 2, suggest calculating a numerical value for the similarity between two sets of clusterings in order to determine whether any changes took place. For complex datasets, we argue that understanding the nature of the change is even more important. We demonstrate this here by comparing our method's results to the following measures:
Measures based on Pair Counting: *RI*: Rand index [31]. *AR*: adjusted Rand index [17]. *MIRKIN*: Mirkin's index [5]. *HI*: Hubert's index [16].
Information Theoretic based Measures:[a] *MI*, mutual information; *NMI*, normalized mutual information; *AMI*, adjusted mutual information [35].
Probability distribution based Measure: *K-L*, the Kullback-Leibler divergence. This is a measure of the dissimilarity between two completely determined probability distributions. $KL\_p = KL(p||q)$,
$KL\_q = KL(q||p)$.
Distance measure based on spatial information: *CDistance*: Comparison function clustering distance [8] [b].

We created a simple synthetic dataset in two dimensions consisting of two clusters, where each cluster has 500 points. We refer to this dataset shown in Figure 10 as the 'reference data'. We also generated 11 datasets drawn from similar distributions as the 'reference data' but with certain changes. The changes made to the distribution represent some of the changes that may occur in the cluster's structure in real scenarios.

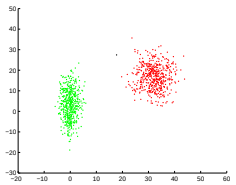


Fig. 10. Reference data

The changes as well as the results of the algorithms are presented in Tables 4& 5.

[a]Code implementing MI NMI and AMI available at http://ee.unsw.edu.au/∼nguyenv/Software.htm.
[b]Code implementing CDistance is available at http://biocomp.wisc.edu/data.

The descriptions of the qualitative results of the TSMSC algorithm use the following symbols: R, the reference dataset; N, the new dataset; $\_CX$, points that belong to cluster X but are not on its boundary; CX_b, points that belong to cluster X's boundary.

The following changes were made to the distribution of the data:

- **Ex1**: No change. The data was generated from the same distribution.
- **Ex2**: The variance of the y coordinate of the points belonging to the green cluster was increased.
- **Ex3**: The variances of the y and x coordinates of the points belonging to the red cluster were decreased.
- **Ex4**: The variances of the y and x coordinates of the points belonging to the red cluster were increased.
- **Ex5**: The green cluster was split into two clusters.
- **Ex6**: The red cluster was split into two clusters.
- **Ex7**: The number of points has changed and a new cluster has been formed.
- **Ex8**: A new cluster was formed with the same number of points.
- **Ex9**: The mean of the green cluster was changed (shifting).
- **Ex10**: The red cluster was caused to disintegrate.
- **Ex11**: The ratio between the number of points in the red cluster and the green cluster was changed.

It is obvious that the methods based on pair counting (e.g., RI and AR) and those based on information theory are not, by definition, appropriate for different data sets or even different samples from the same distribution. In order to be able to apply those methods, the points were ordered such that the cluster's identity changed as little as possible between datasets. In contrast, CDistance overcomes this constraint and can alert for changes between different datasets but does not characterize them.

The is the main problem with all of these methods is that even though they are able in most cases to detect that there was a change in the distribution they are not able to describe what the change was. For example, in Ex3 and EX4 (Table 4) the CDistance succeeded in alerting that changes occurred but assigned them similar values (0.102 and 0.1009) despite the changes being of opposite types (i.e., an increase vs. a decrease in the variance). Most methods are not able to deal with the case when the number of points in the dataset has changed (Ex7).

The KL divergence measures differences between densities. Like the other methods, it produces a numerical value which represents the difference between the distributions (without giving an explanation to the nature of the difference). It is therefore able to detect, for example in Ex11 (Table 5), changes in the proportion of points in the different clusters, while TSMSC is oblivious to such changes.

This is not a major shortcoming of TSMSC since one of its main applications is in the transductive transfer learning setting, where it is required to distinguish whether changes in cluster structure or cluster shape occurred. This information

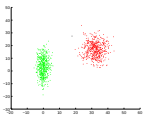
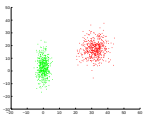
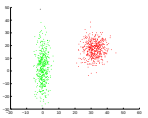
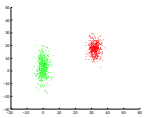
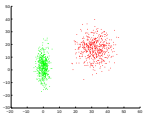
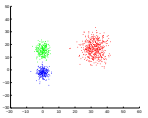
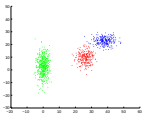
<table>
<tr><td>Reference Clustering
 
Technique Name</td><td>Ex1
  </td><td>Ex2
  </td><td>Ex3
  </td></tr>
<tr><td>RI
AR
MIRKIN
HI
MI
NMI
AMI
KL
CDistance</td><td>0.998
0.999
0.0009
0.998
0.6931
0.9897
0.9948
0.0598 , 0.0410
0.0778</td><td>0.996
0.998
0.002
0.996
0.6931
0.9897
0.9897
0.4369 , 0.1462
0.1419</td><td>0.998
0.999
0.0009
0.998
0.6931
0.9897
0.9948
0.1093 , 0.2690
0.102</td></tr>
<tr><td>TSMSC</td><td>
<table>
<tr><th>C</th><th>0</th><th>1</th><th>1-b</th><th>2</th><th>2-b</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>479</td><td>4</td><td>0</td><td>0</td></tr>
<tr><td>1-b</td><td>0</td><td>1</td><td>16</td><td>0</td><td>0</td></tr>
<tr><td>2</td><td>0</td><td>0</td><td>0</td><td>471</td><td>11</td></tr>
<tr><td>2-b</td><td>1</td><td>0</td><td>0</td><td>0</td><td>17</td></tr>
</table>
No change</td><td>
<table>
<tr><th>C</th><th>0</th><th>1</th><th>1-b</th><th>2</th><th>2-b</th></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>384</td><td>98</td><td>0</td><td>0</td></tr>
<tr><td>1-b</td><td>13</td><td>0</td><td>4</td><td>0</td><td>0</td></tr>
<tr><td>2</td><td>0</td><td>0</td><td>0</td><td>480</td><td>2</td></tr>
<tr><td>2-b</td><td>1</td><td>0</td><td>0</td><td>2</td><td>15</td></tr>
</table>
Part of N_C1 converged to R_C1_b indicating that N_C1 changed its distribution</td><td>
<table>
<tr><th>C</th><th>0</th><th>1</th><th>1-b</th><th>2</th><th>2-b</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>472</td><td>12</td><td>0</td><td>0</td></tr>
<tr><td>1-b</td><td>0</td><td>0</td><td>16</td><td>0</td><td>0</td></tr>
<tr><td>2</td><td>0</td><td>0</td><td>0</td><td>483</td><td>0</td></tr>
<tr><td>2-b</td><td>0</td><td>0</td><td>0</td><td>17</td><td>0</td></tr>
</table>
N_C2_b converged to R_C2 indicating that R_C2 changed its distribution</td></tr>
<tr><td>Technique Name</td><td>Ex4
  </td><td>Ex5
  </td><td>Ex6
  </td></tr>
<tr><td>RI
AR
MIRKIN
HI
MI
NMI
AMI
KL
CDistance</td><td>0.998
0.999
0.0009
0.998
0.6931
0.9897
0.9948
0.1762 , 0.0774
0.10009</td><td>0.7476
0.8739
0.1261
0.7478
0.6931
0.666
0.8123
0.2391 , 0.2211
0.2535</td><td>0.7496
0.8749
0.1251
0.7498
0.6938
0.6667
0.8131
0.5767 , 0.4863
0.2647</td></tr>
<tr><td></td><td>
<table>
<tr><th>C</th><th>0</th><th>1</th><th>1-b</th><th>2</th><th>2-b</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>477</td><td>7</td><td>0</td><td>0</td></tr>
<tr><td>1-b</td><td>0</td><td>1</td><td>15</td><td>0</td><td>0</td></tr>
<tr><td>2</td><td>0</td><td>0</td><td>0</td><td>423</td><td>61</td></tr>
<tr><td>2-b</td><td>4</td><td>0</td><td>0</td><td>0</td><td>12</td></tr>
</table>
Part of N_C2 converged to R_C2_b indicating that N_C2 changed its distribution</td><td>
<table>
<tr><th>C</th><th>0</th><th>1</th><th>1-b</th><th>2</th><th>2-b</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>162</td><td>58</td><td>0</td><td>0</td></tr>
<tr><td>1-b</td><td>0</td><td>25</td><td>4</td><td>0</td><td>0</td></tr>
<tr><td>2</td><td>0</td><td>233</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>2-b</td><td>0</td><td>16</td><td>2</td><td>0</td><td>0</td></tr>
<tr><td>3</td><td>0</td><td>0</td><td>0</td><td>467</td><td>7</td></tr>
<tr><td>3-b</td><td>0</td><td>0</td><td>0</td><td>3</td><td>14</td></tr>
</table>
N_C1 & N_C2 converged to R_C1 indicating that R_C1 split</td><td>
<table>
<tr><th>C</th><th>0</th><th>1</th><th>1-b</th><th>2</th><th>2-b</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>483</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1-b</td><td>0</td><td>1</td><td>15</td><td>0</td><td>0</td></tr>
<tr><td>2</td><td>0</td><td>0</td><td>0</td><td>203</td><td>33</td></tr>
<tr><td>2-b</td><td>0</td><td>0</td><td>0</td><td>12</td><td>4</td></tr>
<tr><td>3</td><td>0</td><td>0</td><td>0</td><td>165</td><td>59</td></tr>
<tr><td>3-b</td><td>2</td><td>0</td><td>0</td><td>18</td><td>4</td></tr>
</table>
N_C2 & N_C3 converged to R_C2 indicating that R_C2 split</td></tr>
</table>

Table 4. Comparison to other works: **Ex1**: No change. The data was generated from the same distribution; **Ex2**: The variance of the y coordinate of the points belonging to the green cluster was increased; **Ex3**: The variances of the y and x coordinates of the points belonging to the red cluster were decreased; **Ex4**: The variances of the y and x coordinates of the points belonging to the red cluster were increased; **Ex5**: The green cluster was split into two clusters; **Ex6**: The red cluster was split into two clusters.

can then be used to decide whether a classifier trained on the first dataset can be used to classify points from the second dataset. Detecting changes in the ratio of the number of points between clusters is not important for this task, although this can be easily provided by analyzing the differences between the number of the

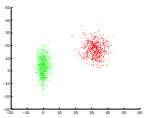
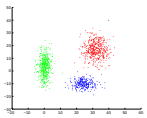
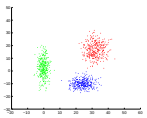
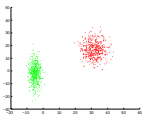
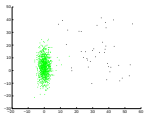
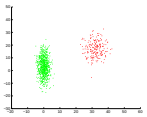
<table>
<tr><td>Reference Clustering
 
Technique Name</td><td>Ex7
  </td><td>Ex8
  </td><td>Ex9
  </td></tr>
<tr><td>RI
AR
MIRKIN
HI
MI
NMI
AMI
KL
CDistance</td><td>NaN
NaN
NaN
NaN
NaN
NaN
NaN
3.6157 , 0.4034
0.3648</td><td>0.4453
0.7230
0.277
0.4459
0.4639
0.4212
0.5289
5.3676 , 0.6372
0.4733</td><td>0.9980
0.999
0.0009
0.998
0.6931
0.9897
0.9948
1.1991 , 1.1944
0.2174</td></tr>
<tr><td>TSMSC</td><td>
<table>
<tr><th>C</th><th>0</th><th>1</th><th>1-b</th><th>2</th><th>2-b</th></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>475</td><td>8</td><td>0</td><td>0</td></tr>
<tr><td>1-b</td><td>0</td><td>1</td><td>17</td><td>0</td><td>0</td></tr>
<tr><td>2</td><td>0</td><td>0</td><td>0</td><td>477</td><td>5</td></tr>
<tr><td>2-b</td><td>0</td><td>0</td><td>0</td><td>0</td><td>17</td></tr>
<tr><td>3</td><td>290</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>3-b</td><td>9</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>
N_C3 converged to cluster_0,
N_C3 is a new cluster</td><td>
<table>
<tr><th>C</th><th>0</th><th>1</th><th>1-b</th><th>2</th><th>2-b</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>315</td><td>7</td><td>0</td><td>0</td></tr>
<tr><td>1-b</td><td>0</td><td>2</td><td>9</td><td>0</td><td>0</td></tr>
<tr><td>2</td><td>0</td><td>0</td><td>0</td><td>320</td><td>2</td></tr>
<tr><td>2-b</td><td>0</td><td>0</td><td>0</td><td>2</td><td>8</td></tr>
<tr><td>3</td><td>323</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>3-b</td><td>9</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
</table>
N_C3 converged to cluster_0,
N_C3 is a new cluster</td><td>
<table>
<tr><th>C</th><th>0</th><th>1</th><th>1-b</th><th>2</th><th>2-b</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>298</td><td>185</td><td>0</td><td>0</td></tr>
<tr><td>1-b</td><td>1</td><td>4</td><td>11</td><td>0</td><td>0</td></tr>
<tr><td>2</td><td>0</td><td>0</td><td>0</td><td>470</td><td>14</td></tr>
<tr><td>2-b</td><td>0</td><td>0</td><td>0</td><td>4</td><td>12</td></tr>
</table>
Part of N_C1 converged to R_C1_b
indicating that N_C1 changed
its distribution</td></tr>
<tr><td>Technique Name</td><td>Ex10
  </td><td>Ex11
  </td><td></td></tr>
<tr><td>RI
AR
MIRKIN
HI
MI
NMI
AMI
KL
CDistance</td><td>0.0083
0.5029
0.4971
0.0058
0.0331
0.0463
0.0915
1.0821 , 2.2308
0.9254</td><td>0.1589
0.579
0.421
0.158
0.1644
0.2329
0.276
0.3139 , 0.3876
0.6218</td><td></td></tr>
<tr><td></td><td>
<table>
<tr><th>C</th><th>0</th><th>1</th><th>1-b</th><th>2</th><th>2-b</th></tr>
<tr><td>0</td><td>20</td><td>0</td><td>0</td><td>14</td><td>12</td></tr>
<tr><td>1</td><td>0</td><td>903</td><td>22</td><td>0</td><td>0</td></tr>
<tr><td>1-b</td><td>0</td><td>0</td><td>28</td><td>0</td><td>1</td></tr>
</table>
No cluster was mapped
to R_C2 indicating that
cluster R_C2 faded away</td><td>
<table>
<tr><th>C</th><th>0</th><th>1</th><th>1-b</th><th>2</th><th>2-b</th></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>769</td><td>5</td><td>0</td><td>0</td></tr>
<tr><td>1-b</td><td>0</td><td>1</td><td>25</td><td>0</td><td>0</td></tr>
<tr><td>2</td><td>0</td><td>0</td><td>0</td><td>191</td><td>0</td></tr>
<tr><td>2-b</td><td>0</td><td>0</td><td>0</td><td>5</td><td>3</td></tr>
</table>
No change</td><td></td></tr>
</table>

Table 5. Comparison to other work: **Ex7**: The number of points has changed and a new cluster has been formed; **Ex8**: A new cluster was formed with the same number of points; **Ex9**: The mean of the green cluster was changed (shifting); **Ex10**: The red cluster was caused to disintegrate; **Ex11**: The ratio between the number of points in the red cluster and the green cluster was changed.

points in corresponding clusters.

The main difference between TSMSC and the other methods is its ability to produce a qualitative description of the changes between the datasets. For each of the experiments the confusion matrix and its interpretation is provided. In complex datasets such an analysis is crucial to help the domain expert understand the changes that have occurred.

6.3. *Experiments on real stereo images*

When performing experiments on real data, the data should be complex but the clustering results should be easy for the user to evaluate. Moreover, there should be some high level matching result which could easily be tested by the user although it is unknown to the algorithm.

For these reasons we tested our algorithm on a pair of real color images known as stereo pairs (two images of a scene taken by a camera from two viewpoints). These datasets satisfy our requirements: the data sets are sufficiently large and complex, and the results obtained by the $TSMSC$ algorithm can be easily viewed and evaluated. Moreover, as a result of horizontal motion of the camera, objects in the scene move horizontally between the images a distance that is inversely proportional to their depth. This distance is known as the disparity.

Working with images is not an easy task. Each pixel in the image is represented by its two image coordinates (X, Y) and its RGB color values, yielding a 5D dataset. A commonly used system for analyzing images is the Edge Detection and Image Segmentation (*EDISON*) System. This program implements the mean shift image segmentation algorithm described in [7]. The user is asked to give the algorithm values for two bandwidths, one for the spatial domain h_{XY} (the image coordinates) and the other for the range domain h_{RGB} (the RGB values). The output of this program is a clustered image. Each cluster is assigned a color, (i.e., points in the same cluster have the same color). EDISON is an interactive program which is able to segment an image in several seconds. Using $EDISON$ as our model, we implemented a variant of $TSMSC$ for color images.

The main difference between this variant of $TSMSC$ and the basic algorithm described in the previous sections is that this variant requires that two bandwidths be given to the algorithm. To deal with this case we simply build a two-dimensional table in which the vertical axis is for different values of h_{xy} and for each row we execute the $finding\_h$ algorithm (Algorithm 1) on h_{RGB}. We then use the results to find an appropriate and stable bandwidth for RGB. The same can be done in cases where more than two bandwidths are required.

6.3.1. *Experimental results*

In addition to the aforementioned disparity, we also expect that some clusters will merge or split as naturally happens when two images are taken from two viewpoints.

The images were taken from the Middlebury College Stereo Vision Research Web Page [27]. This Web site is intended for comparing stereo algorithms. Figure 11 shows the image pair on which we tested our algorithm.

The segmentation partitioned image 1 into 503 clusters and image 2 into 530 clusters. The size of the confusion matrix is thus 530×530. We therefore present only a small slice of it in Table 6. The vertical axis represents clusters from image 2 and the horizontal axis represents clusters from image 1. Each row shows the number of pixels from one cluster in image 2 that converged to each cluster of

Image 1 Image 2 Image 1 segmentation Image 2 segmentation

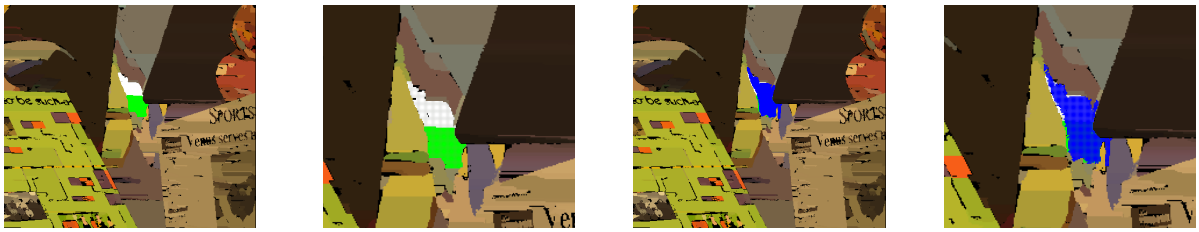
Fig. 11. Images 1, 2 and the segmentation results with $h_{XY} = 8$ and $h_{RGB} = 30$

image 1.

	0	...	**17**	...	**60**	...	**66**	...	**70**	...	**97**	...	**147**	...	**178**	...
...																
81			5				63		680		15					
...																
117	15				35				803				2		8	
...																

Table 6. A slice from the confusion matrix of the images. The columns represent image 1 clusters while the rows represent image 2 clusters.

An analysis of the rows shows that cluster 81 is mapped to cluster 70, and an analysis of the columns shows that cluster 70 is the merger of clusters 81 and 117. Figure 12 shows clusters 81 and 117 from image 2 and cluster 70 from image 1.



Im 2: Clusters 81 & 117 Zoom in Im 1: Cluster 70 Zoom in

Fig. 12. Clusters 81 and 117, mapped to cluster 70

As mentioned above, the main difference between the images is the disparity between segments. Consider, for example, the boundary between two segments A and B. The X coordinate changes while the Y and RGB components stay the same. Figure 13 illustrates the different possible cases for pixels lying close to the boundary between clusters which underwent movement. We will concentrate on pixel $\mathbf{c}$. This pixel, which belongs in the first image to segment G, will have coordinates

(X, Y, RGB_G) while in the second image it will have coordinates (X, Y, RGB_R). Assuming that the colors are different enough, this pixel will not belong to any segment when the dataset of the first image is used. It will not belong to segment G because $d(RGB_G, RGB_R) > h_{RGB}$ and it will not belong to segment R because the points belonging to segment R in the first image are not close to it in the spatial domain. Thus, it will be associated with *cluster*_0. These pixels can be used to estimate the disparity of the cluster.

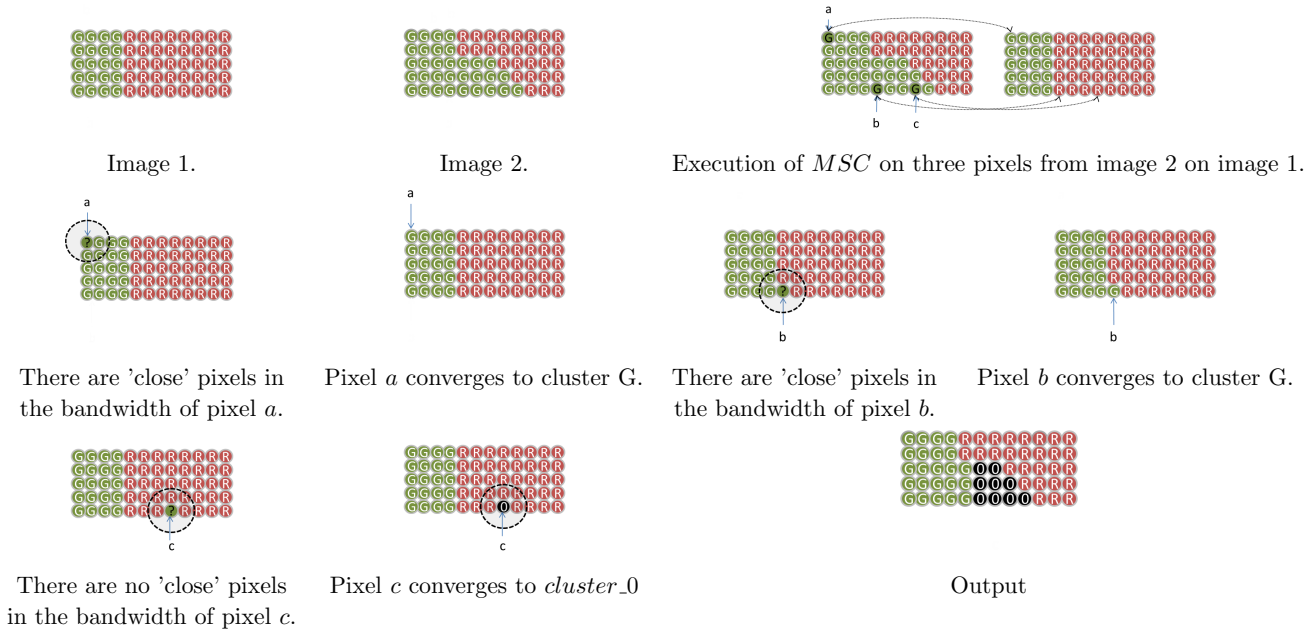
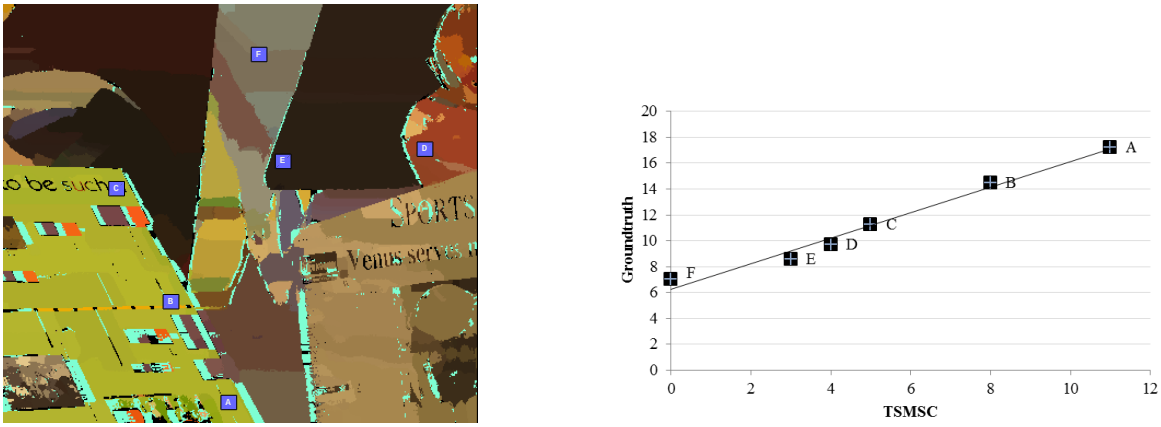


Fig. 13. TSMSC output as a result of segment motion

Figure 14(a) shows the pixels that belong to a cluster in image 2 and that $TSMSC$ associated those pixels with *cluster*_0 on image 1. Those pixels are colored in cyan. We focus on six segments, marked A, B, C, D, E, and F. In these examples a wide range of disparity values can be seen (Recall that the disparity is inversely proportional to the depth of the object.) Segments A, B, and C lie on a planar surface inclining away from the camera. This is evident because the disparity values decrease continuously. The disparities of segments C and D are quite close and therefore so are their depths. E is further away. No disparity is measured for segment F, and thus it is furthest from the camera.

Figure 14(b) plots the ground truth disparity provided by the Middlebury Web site as a function of the width of the six *cluster*_0 regions. The difference between the values is close to constant (between 5.5 and 7 pixels). As Figure 13 shows, this is because pixels close to the boundary between the two clusters (e.g. pixel **b**) are



(a) (b)

Fig. 14. Stereo results: (a) The disparity map of image 2 (b) The ground-truth disparity map vs. TSMSC results

not associated with *cluster*_0. Thus the disparity can be estimated as the width of the *cluster*_0, to which a constant (which is a function of h_{XY}) is added.

In conclusion, this example shows how the algorithm is able to correctly analyze the data and even produce an initial estimate of the disparity.

7. Conclusions

In this paper we proposed a dataset change analysis method based on the mean shift clustering algorithm. Our non-parametric algorithm makes no assumptions on the structure of the data (besides that it can be clustered). The method is in essence density based. However, it uses mean shift to analyze changes in cluster structure rather than changes in density alone, yielding more meaningful results. It first recovers a stable cluster structure in both datasets. Each point from one of the datasets is associated also with a cluster from the other dataset. The resulting confusion matrix serves as the basis for our analysis of the changes between the datasets. The changes are not given as numerical values but descriptively.

The algorithm was also used as the basis for a transductive transfer learning method for automatically training a classifier on the new dataset using the old classifier and the results of the algorithm.

The main strength of the algorithm is that it can be applied to datasets drawn from general distributions but this is also its main weakness. If it can be assumed for

example that the distribution can be approximated as a Gaussian mixture model, where each cluster is approximated as drawn from a Gaussian distribution, the GMM clustering method could be used. In addition, methods for comparing samples from Gaussian distributions can be used to test whether or not there has been a change from one dataset to the other. This can probably be done more efficiently and with smaller sample sizes then is required by our algorithm.

Future work will focus on developing efficient methods for detecting whether changes have occurred using only a small subset of points selected from the new dataset, and applying the algorithm to the two complete data sets only when a change has been detected.

References

1. Fabrizio Angiulli and Clara Pizzuti. Outlier mining in large high-dimensional data sets. *IEEE Trans. on Knowl. and Data Eng.*, 17(2):203–215, February 2005.
2. Eric Bae, James Bailey, and Guozhu Dong. A clustering comparison measure using density profiles and its application to the discovery of alternate clusterings. *Data Min. Knowl. Discov.*, 21:427–471, November 2010.
3. Vic Barnett and Toby Lewis. *Outliers in Statistical Data*. Wiley Series in Probability & Statistics. Wiley and Sons, New York, NY, USA, April 1994.
4. Michèle Basseville and Igor V. Nikiforov. *Detection of Abrupt Changes: Theory and Application*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.
5. L.B. Chernyi B.G. Mirkin. Measurement of the distance between distinct partitions of a finite set of objects. *Automation and Remote Control*, 31(5):786–792, 1970.
6. Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: identifying density-based local outliers. *SIGMOD Record*, 29(2):93–104, May 2000.
7. C. Christoudias, B. Georgescu, and P. Meer. Synergism in low level vision. In *Proceedings of International Conference on Pattern Recognition*, pages 150–155, 2002.
8. Michael H. Coen, M. Hidayath Ansari, and Nathanael Fillmore. Comparing clusterings in space. In *ICML '10*, 2010.
9. D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Patt. Anal. Mach. Intell.*, 24(5):603–619, 2002.
10. Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Trans. Patt. Anal. Mach. Intell.*, 25(5):564–577, 2003.
11. Acuna Edga and Caroline Rodriguez. Meta analysis study of outlier detection methods in classification. In *Proceedings IPSI, Venice*, 2004.
12. João Gama. *Knowledge Discovery from Data Streams*. Boca Raton: Chapman & Hall\CRC, New York, NY, USA, 2010.
13. João Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *SBIA*, volume 3171 of *LNCS*, pages 286–295, Berlin, Germany, 2004. Springer-Verlag.
14. B. Georgescu, I. Shimshoni, and P. Meer. Mean shift based clustering in high dimensions: a texture classification example. In *Proc. Int. Conf. Comp. Vision*, pages 456–463, 2003.
15. A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. Int. Conf. on Very Large Data Bases*, pages 518–529, 1999.
16. Lawrence Hubert. Nominal scale response agreement as a generalized correlation. *British Journal of Mathematical and Statistical Psychology*, 30(1):98–103, 1977.

17. Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, December 1985.
18. Asa B. Hur, Andre Elisseeff, and Isabelle Guyon. A stability based method for discovering structure in clustered data. In *Pacific Symposium on Biocomputing*, pages 6–17, 2002.
19. P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proc. Symp. on Theory of Computing*, pages 604–613, 1998.
20. Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *VLDB*, pages 180–191. VLDB Endowment, 2004.
21. Edwin M. Knorr, Raymond T. Ng, and Vladimir Tucakov. Distance-based outliers: algorithms and applications. *The VLDB Journal*, 8(3):237–253, 2000.
22. Bjornar Larsen and Chinatsu Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 16–22, 1999.
23. Antonio Loureiro, Luis Torgo, and Carlos Soares. Outlier detection using clustering methods: a data cleaning application. In *Proceedings of the Data Mining for Business Workshop*, 2004.
24. Junshui Ma and Simon Perkins. Time-series novelty detection using one-class support vector machines. In *Proceedings of the International Joint Conference on Neural Networks*, volume 3, pages 1741–1745. IEEE, 2003.
25. Marina Meilă. Comparing clusterings. Technical Report 418, UW Statistics Technical Report, 2003. http://academic.uprm.edu/eacuna/paperout.pdf.
26. Marina Meilă. Comparing clusterings: an axiomatic view. In *Proceedings of the International Conference on Machine Learning*, pages 577–584, 2005.
27. Middlebury College. Stereo vision research web page, 2002. http://vision.middlebury.edu/stereo/.
28. E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.
29. A Kai Qin and Ponnuthurai N Suganthan. Robust growing neural gas algorithm with application in cluster analysis. *Neural Networks*, 17(8):1135–1148, 2004.
30. Parasaran Raman, Jeff M. Phillips, and Suresh Venkatasubramanian. Spatially-aware comparison and consensus for clusterings. *CoRR*, abs/1102.0026, 2011. http://arxiv.org/abs/1102.0026.
31. W. M. Rand. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66:846–850, 1971.
32. P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. John Wiley & Sons, Inc., New York, NY, USA, 1987.
33. Raquel Sebastiao and Joao Gama. A study on change detection methods. In *New Trends in Artificial Intelligence, Proceedings of the 14th Portuguese Conference on Artificial Intelligence (EPIA 2009)*, pages 353–364, Aveiro, Portugal, 2009.
34. J. Tukey. Mathematics and the picturing of data. In *Int. Congress of Mathematicians*, pages 2:523–531, 1975.
35. Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *J. Mach. Learn. Res.*, 11:2837–2854, December 2010.
36. Ji Zhang and Hai Wang. Detecting outlying subspaces for high-dimensional data: the new task, algorithms, and performance. *Knowl. Inf. Syst.*, 10(3):333–355, October 2006.