# TASOCNN: A Topology Adaptive Self-Organizing Circular Neural Network and its Application to Color Segmentation

Anandarup Roy
CVPR Unit
Indian Statistical Institute
203 B. T. Road
Kolkata 700108, India
roy.anadarup@gmail.com

Swapan Kumar Parui*
CVPR Unit
Indian Statistical Institute
203 B. T. Road
Kolkata 700108, India
swapan@isical.ac.in

Utpal Roy
DCSS
Visva-Bharati University
Santiniketan
Birbhum 731235, India
roy.utpal@gmail.com

## ABSTRACT

Topology adaptive neural networks are popular because of its capability to adopt the underlaying topology of data. In this paper we develop a topology adaptive self organizing circular neural network (TASOCNN) model for circular-linear data sampled from an unit disk. The basic framework uses the TASONN procedure of Datta et. al. [8]. The update rules and the distance measure are reformulated with the inclusion of directional information. In the iterative TASOCNN process, we create/update edges between any two winner processors. These edges are weighted, hence finally a weighted processor graph is created concerning processors as vertices. By removing possible inter-cluster edges from the connected components of this processor graph, a set of subgraphs can be obtained. For this purpose we use a cost function based on edge length and strength. Many of these subgraphs become close to one another. These close subgraphs are merged. Finally, each subgraph represents a cluster in the data. We apply TASOCNN for color based image segmentation. TASOCNN is constructed in the hue-saturation space. The Berkeley segmentation dataset is used to present the results. An evaluation is made by means of probabilistic rand index. Our experiments reveals satisfactory outcome of TASOCNN.

## 1. INTRODUCTION

Color image segmentation [1] is becoming increasingly important in many applications since color images are now easily available and can provide more information than gray level images. In this article we discuss color based segmentation of images ("Color segmentation" in short). Such color based segmentation processes consider only the color information for image segmentation. In a natural scene, an object of homogenous color may have different shades (or brightness) due to change in light. If the effects of shade or brightness are not ignored, the object may be segmented into several parts according to brightness. This situation leads us to the study of color based segmentation methods. In this context, the hue-saturation-value (HSV) color system becomes more suitable than the RGB system, since it separates out color information. Typically, the hue and saturation components together represent corresponding pixel color. Color image segmentation methods can be roughly categorized into three groups [2]: (1) clustering based segmentation, (2) edge or contour detection based segmentation and (3) region or area extraction based segmentation. In this article, we apply clustering based segmentation approach. The basic idea behind this approach is to directly cluster the pixels in a certain color space by employing some clustering algorithms. Generally clustering approaches may be parametric or non-parametric. Popular non-parametric approaches include mean-shift and normalized-cut procedures [3] which are used quite widely in vision community. In addition, self organizing structures including Self-Organizing Map (SOM) [4] and Neural/Growing Neural Gas (NGN/GNGN) [5], are also studied for clustering. The SOM projects input space on processors of lower dimensional grid, which can be used to explore properties of the data. When the number of processors becomes large, several clusters may be identified by grouping similar processors. Vesanto et. al [6] discussed hierarchical clustering process to group similar processors. Concerning color image segmentation, Jiang et. al [7] proposed clustering an image with SOM and merging "scattered blocks" [7] to obtain different segments. Datta et. al [8] proposed a topology adaptive self organizing neural network (TASONN) to generate the skeleton of a pattern. This model is a dynamic network that grows over time. Their study showed better performance of TASONN compare to SOM, NGN and GNGN. The discussions in [8] yields the fact that the NGN and GNGN model use an edge destruction procedure to eliminate certain non-relevant edges between processors. This may cause a sudden loss of information. The TASONN model, on the other hand, use a strength measure with each of the edges. The strength increases for the relevant processors and decreases for others. Several other advantages of TASONN are discussed in [8].

In this paper we develop a self organizing network for circular-linear data. This network is termed as TASOCNN. Its basic structure is motivated by TASONN of Datta et. al. The update rules are modified to make the processor

*Corresponding author


*ICVGIP* '10, December 12-15, 2010, Chennai, India

weight vectors get updated on an unit disk. A circular-linear distance is defined to find the neighborhood processors. Though we present TASOCNN for two dimensional circular-linear data, the same framework can be easily extended to multidimensional data. We construct a processor graph by creating edges between neighborhood processors. Several connected components are extracted from this graph by eliminating certain undesirable edges. During this deletion process we apply a cost function on edge lengths and strengths. We realize different clusters as different connected components of the processor graph. We perform experiments with Berkeley segmentation dataset which is a benchmark dataset containing ground truth segmentations. The comparison is done using probabilistic rand index that gives a score indicating performance.

## 2. TASOCNN: DESCRIPTION AND ALGORITHM

Let the input patterns be $\mathcal{X} = \{X_1, \ldots, X_m\}$, in which, each $X_l$ is an input pattern on an unit disk $\mathcal{D}$. Each $X_l = (\theta_l, s_l)$, $(0 \leq \theta_i < 2\pi, 0 \leq s_i \leq 1; i = 1, \ldots, m)$ is described by the angular component $\theta_l$ and the linear component $s_l$. We denote $\Pi = \{\pi_1, \ldots, \pi_{n(r)}\}$ be the set of processors where $n(r)$ is the number of processors at iteration $r$. Let the neighborhood of a processor $\pi_i$ be the set $N_i$ defined as $N_i = \{\pi_p | \pi_p \text{ is connected to } \pi_i\}$. The processor $\pi_i$ is a tuple $(\theta_i, s_i)$ describing its position on $\mathcal{D}$. An edge between two processors $\pi_i$ and $\pi_j$ is denoted by $L_{ij}$ and is undirected. Next we give some definitions concerning the TASOCNN procedure.

**Definition 1.** By *sensitive region* of a processor $\pi_i$ we mean a circle of a given radius centered at the position of $\pi_i$. The radius is called the *sensitive level*.

**Definition 2.** A processor is said to *win* if it is the nearest to the concerned input pattern among all other processors. That processor is termed as the *winner* processor. The second nearest processor becomes the *second winner*.

**Definition 3.** One pass through the input vectors is said as a *phase*. Several phases may occur in one iteration of the algorithm.

Several control parameters of the algorithm are defined as follows. These parameters are given as inputs to the algorithm.

**Sensitive level** is denoted by $\tau$ and used to create new processors.

**Phase Completion threshold** is denoted by $\epsilon$. The difference between previous and current processor set is tested against $\epsilon$ to complete a phase.

**Learning rates** are denoted by $\alpha_1$ and $\alpha_2$ $(0 < \alpha_2 \leq \alpha_1 < 1)$, for first and second winners respectively. Learning rates are variable unlike $\tau$ and $\epsilon$.

Considering both the linear and circular aspects of the data, the distance between two input patterns $X_1 = (\theta_1, s_1)$ and $X_2 = (\theta_2, s_2)$ on $\mathcal{D}$, is defined as:

$$d(X_1, X_2) = \frac{\min(|\theta_1 - \theta_2|, 2\pi - |\theta_1 - \theta_2|)}{\pi} + |s_1 - s_2|. \quad (1)$$

In this formulation, the first component is the normalized circular distance while the second component is the linear distance. The winners are detected using this distance measure. In TASOCNN model, for each input pattern, we consider the winner and the second winner processors and update them. Let, after presenting an input $X = (\theta, s)$ at time $t$, the winner processor be denoted by $\pi_u = (\theta_u, s_u)$ and the second winner by $\pi_v = (\theta_v, s_v)$. The processors are updated in both circular and linear directions. Let

$$\begin{aligned} S_u &= \alpha_1^{(t)} \sin\theta + (1 - \alpha_1^{(t)}) \sin\theta_u^{(t)} \text{ and} \\ C_u &= \alpha_1^{(t)} \cos\theta + (1 - \alpha_1^{(t)}) \cos\theta_u^{(t)}. \end{aligned}$$

In other words, $S_u$ and $C_u$ are the sine and cosine components of the resultant vector after summing up $\theta$ and $\theta_u^{(t)}$. Then the circular update is given by

$$\theta_u^{(t+1)} = \tan^{-1}\left(\frac{S_u}{C_u}\right). \quad (2)$$

Note arctan should take values in the range $[0, 2\pi)$. The update in $\theta_v^{(t)}$ can be computed in a similar way using update rate $\alpha_2$. The updates in the linear components $s_u$ of $\pi_u$ and $s_v$ of $\pi_v$ with respect to the linear component $s$ of $X$, are given as

$$s_u^{(t+1)} = s_u^{(t)} + \alpha_1^{(t)}\{s - s_u^{(t)}\} \quad (3)$$

$$s_v^{(t+1)} = s_v^{(t)} + \alpha_2^{(t)}\{s - s_v^{(t)}\}. \quad (4)$$

At a time instant $t$ the winners $\pi_u$ and $\pi_v$ are joined by an edge $L_{uv}$. A strength $\beta_{uv}$ is associated with every such edges. Initially all strengths are zero. During iterations the strengths are updated as follows.

$$\beta_{uv}^{(t+1)} = \beta_{uv}^{(t)} + \frac{1}{t+1}\left(1 - \beta_{uv}^{(t)}\right). \quad (5)$$

For all other edges,

$$\beta_{ij}^{(t+1)} = \frac{t}{t+1}\beta_{ij}^{(t)}, \ (i,j) \neq (u,v). \quad (6)$$

Based on the above descriptions we now detail the precise steps of the TASOCNN algorithm.

**Initialization:** Initially $\Pi = \phi$. The control parameters– $\tau$, $\alpha_1^{(0)}$, $\alpha_2^{(0)}$ and $\epsilon$ are given by the user. The input pattern $X_t$, $(t = 1)$, is presented for processing.

**Processor creation:** A new processor $\pi_{new}$ is created if $\Pi = \phi$ or $X_t$ is not inside the sensitive region of any processors. The new processor is placed at the position of $X_t$. Also $\Pi$ is redefined as $\Pi = \Pi \cup \{\pi_{new}\}$.

**Winner detection:** The winners are defined as in definition 3 and detected using equation 1. The second winner may not exists.

**Construct or update edge:** If the second winners does not exist go to next step, otherwise join the first and the second winners ($\pi_u$ and $\pi_v$) by an edge $L_{uv}$. If the edge is newly created then its strength $\beta_{uv}$ is zero. Otherwise $\beta_{uv}$ of $L_{uv}$ is updated according to equation 5. All other strengths are updated using equation 6.

**Processor update:** The winner processors are updated by applying the update rule with respect to the data pattern $X_t$. If the second winner doesn't exists only the first winner is updated.

**Sweep:** Set $t = t + 1$ and thus present the next input pattern. Continue till all the input patterns are presented.

**Phase Completion:** After sweep $t_s$ if no new processor is created and $|\Pi(t_s) - \Pi(t_{s-1})| < \epsilon$ then the current phase is completed. Otherwise start a fresh sweep (i.e. present $\mathcal{X}$ again). Only after a phase the learning rates $\alpha_1$ and $\alpha_2$ are subjected to a decay function. At time $t_s$ the learning rate $\alpha_1^{(t_s)}$ and $\alpha_2^{(t_s)}$ are computed by

$$\alpha_1^{(t_s)} = \frac{\alpha_1^{(0)}}{1 + \frac{t_s}{50}}$$

$$\alpha_2^{(t_s)} = \frac{\alpha_2^{(0)}}{1 + \frac{t_s}{20}}$$

**Processor insertion:** The processors are inserted only after a phase is completed. If the termination condition is not met, a processor is inserted between the two neighborhood processors $\pi_k$ and $\pi_{k'}$ where the edge $L_{kk'}$ between $\pi_k$ and $\pi_{k'}$ has maximum strength. Further start a new iteration with $\mathcal{X}$ and the modified set of processors.

**Output:** After termination, the output of TASOCNN is a processor graph $G(V, E)$ where $V = \Pi$ and $E = \{L_{ij} | \pi_i, \pi_j \in \Pi, \forall i \neq j\}$.

The termination condition is set as follows. Our application of the TASOCNN is to cluster the dataset $\mathcal{X}$. If we consider each processor as one cluster, then a handful of data must be present in the Voronoi polygon of each processor. If a certain number of data samples are not present in the Voronoi polygon of all processor we may conclude that the maximum number of processors are reached and terminate the algorithm.

# 3. PROCESSOR GRAPH CLUSTERING

After completion of TASOCNN iterations, the processor graph formed by joining the processors, represents the clouds in the input data. We assume, the processor graph reflects the properties of the input space. Thus, the clusters found from the processor graph should be close to those found from the data directly. Initially, let the connected components of the graph, correspond to different clusters. We term the edges among the processors of such a connected component as intra-cluster edges. When the clusters are far apart, the connected components are self contained in the sense only intra-cluster edges are present with no inter-cluster connections. Then clustering can be done simply by picking the connected components of the processor graph. Situations become difficult when some clusters are close enough or even overlap. Then the components for two or more clusters are connected by the inter-cluster edges. This is a more general scenario. We even obtained a single connected processor graph in many of our experiments. In this situation clusters can only be found by properly de-linking the components by removing the inter-cluster edges. We here define a cost function that puts a limit when we are removing edges. However, experiments shows, we may remove too many edges, thus obtaining more clusters than necessary. So a graph merging procedure is performed to merge certain connected components i.e. subgraphs obtained after edge removal process. This procedure is based on spectral distance measure between subgraphs. Concerning literatures, our method has a resemblance to graph partitioning based clustering methods. These methods take the data points to represent the vertices of a graph $\mathcal{G}$. An edge is used to join two data points (vertices) if they are neighbors, given a pre-defined neighborhood system. Graph partitioning methods involve partitioning criteria such as the min-cut algorithm [9]. The criteria are used to find mutually exclusive subgraphs by removing edges from $\mathcal{G}$. These subgraphs correspond to the clusters. Here, we can say, the processors can correctly represent the image. In other words, each of the processors can be a representative of the pixels that lie inside its Voronoi polygon. Now considering the processors instead of original image pixels, we may employ a graph based clustering method. The edges are, however, defined during TASOCNN procedure. An edge between two processors has a weight that decides the connection strength of the two corresponding processors. Since the processors at the high density areas of the data cloud win frequently, it turns out that the edges among such processors have more weight than other edges. We could obtain subgraphs by removing the weak edges. These resulting subgraphs may successfully represent the clusters. These weak edges are actually the inter-cluster edges mentioned earlier. This approach is followed by a simple subgraph merging procedure to merge close subgraphs. The processor graph partitioning criterion is based on two basic properties of the edges: the edge strength and the edge length. A large number of edges are removed based on these two properties. Hence, in general the number of subgraphs become larger than necessary. The merging procedure intends to merge "close" subgraphs by applying a "closeness" criteria. In the following subsections, we describe the edge removal and subgraph merging methods.

## 3.1 Inter-cluster edge removal

We, at first, isolate the connected components of the processor graph. Each component is examined separately to remove the inter-cluster edges. For this purpose, we consider the strengths of the edges. Our study reveals, most of the intra-cluster edges have greater strength than the inter-cluster edges. Thus, if we remove edges with less strengths, the inter-cluster edges are expected to be removed. Yet, considering only the edge strength gives insufficient information. Also, it is not easy to find the threshold between less and greater strength edges. Hence, we further consider the lengths of different edges. We observe, the inter-cluster edges having greater length compere to the intra-cluster edges. This is justified, since more processors are inserted in the portion with high data density, i.e. within a certain cluster. According to the insertion process, each insertion shorter the edge length. Thus, if we remove the long edges we may end up to a desired processor graph representing the clusters. Again, considering only the edge lengths may not produce the proper solution. A better approach may be to consider edges along with their strengths. In this context, we observe a relation between edge length and edge strength. The inter-cluster edges are mostly longer and having less strength than the intra-cluster edges. During edge creation a longer edge may be created between two processors far enough. However, the strength of that edge gradually decrease according to edge update process. This is because the chance of the two corresponding processors to become winners is less after inserting more processors. So, the above

relation is justified. Let us denote an edge between $\pi_i$ and $\pi_j$ by $L_{ij} = (l_{ij}, \beta_{ij})$ with length $l_{ij}$ and strength $\beta_{ij}$ (both normalized). The length here is measured in the hue-saturation space. We define a cost function $g$ as:

$$g = \sum_{i,j} l_{ij} - \sum_{i,j} \beta_{ij}. \quad (7)$$

The function $g$ actually defines the effect of strength on the edge lengths. We remove the edges in an increasing order of strengths, so first we delete the edge with smallest strength. Consequently, we hope to remove the longest edge. We do this iteratively until we reached at some pre-defined point. With equation 7 the iterative process can be described as:

$$\begin{aligned} g_0 &= \sum_{i,j} l_{ij} - \sum_{i,j} \beta_{ij} \\ g_{t+1} &= g_t - L_{ij}, \quad \text{s.t } L_{ij} = (l_{ij}, \beta_{ij}), \\ \text{where } \beta_{ij} &< \beta pq, \ \forall (p,q) \neq (i,j). \end{aligned}$$

After the removal of possible inter-cluster edges we end up into a set of $M$ subgraphs $G^M = \{G_i | i = 1, \ldots, M\}$. We consider each subgraphs as a separate cluster. Thus a total of $M$ clusters are obtained with this process. Note, a set of isolated processors may be produced by the edge removal process. We ignore isolated processors and eliminate them from the set of final processors. Another implementation issue arise here. The processor graph, if complete, have $|\Pi|(|\Pi|-1)/2$ edges, which is a large number (of $O(|\Pi|^2)$). In practice, however, the processor graph does not become exactly complete, yet contains a large number of edges. Many of these edges have a minimal strength and removed in the first few iterations of the edge removal process. We take an attempt to rigorously eliminate these edges based on a threshold on minimum edge strength. This speeds up the edge removal process.

## 3.2 Merging subgraphs

After obtaining $M$ subgraphs we may connect each subgraphs to a separate cluster. However, the edge removal procedure tends to produce more subgraphs than the total possible number of clusters. Hence we next design a procedure to merge several subgraphs to obtain a subgraph that originally represents a cluster. The merging procedure is applied to the subgraphs $G_1, \ldots, G_M$ of $G^M$ till a pair is found to merge. In each iteration, the two subgraphs having minimum distance are merged. The distance between two subgraphs $G_i$ and $G_j$ is defined as follows.

$$d(G_i, G_j) = \max_{\pi_l \in G_i, \pi_m \in G_j} d(\pi_i, \pi_j)$$

where, $d(\pi_i, \pi_j)$ denotes the distance between processors $\pi_i$, $\pi_j$ and can be computed using equation 1. In other words the distance between two subgraphs is the maximum distance between all pair of processors from the two subgraphs. This distance and the corresponding merging procedure resemblance to complete linkage principle. Certainly, another way is to define the subgraph distance to be the minimum of the pairwise processor distances. This method is similar to single linkage principle. We apply both the two methods and compare their outcomes by means of an evaluation measure. For both the merging methods, the choice of distance threshold that controls the merging is important. The distance threshold puts a bound on the distance among subgraphs. Too large the threshold results too less number of final subgraphs. In effect we have only a few clusters. On the other hand too small the threshold may results large number of clusters while many of them are undesired. Here, the distance threshold is found empirically.

# 4. EVALUATION OF SEGMENTATION

In order to qualify the segmentation algorithm we have to compare the resulting clusters with some ground-truth segmentations. The evaluation of segmentation has become an issue of interest since ages. As a result, a number of measures have been proposed for an objective evaluation of a segmentation algorithm. In 2003, Martin [10] designed several error measures to quantify the consistency between image segmentations of differing granularities, and used them to compare the results of normalized-cut algorithms to a database of manually segmented images. Recently, Unnikrishnan et. al. [11] proposed a Probabilistic Rand (PR) Index which is a generalization of a classical non-parametric test known as Rand Index [12]. Besides the examples described by Unnikrishnan et. al., more recently, the PR index is applied to evaluate a color segmentation procedure designed by Ilea and Whelan [13]. The PR index allows comparison of a test segmentation with multiple ground-truth images by evaluating the pairwise relationships between pixels. In other words, the PR index measures the agreement between the segmented result and the manually generated ground-truths and takes values in the range $[0, 1]$, where a higher PR value indicates a better match between the segmented result and the ground-truth data. In this study we use the PR index (PRI) to evaluate the segmentation process. Concerning study of image segmentation evaluation, a number of unsupervised methods has described by Zhang et. al. [14]. These unsupervised methods can perform objective evaluation without having human visual comparisons or comparison with a manually-segmented reference image.

# 5. COLOR SEGMENTATION: RESULTS AND DISCUSSIONS

Color segmentation considers only the color information for image segmentation purpose. Color information is presented by hue and saturation values of an HSV image. Hue and saturation values together represent a point on an unit disc, therefore suitable for TASOCNN.

The results are presented on Berkley segmentation dataset [15]. This dataset contains several color images along with human segmentation results. Each image has an unique Id. The images contain at least one distinguishable thing i.e. identifiable object embedded in a natural scene. Some example images are shown in Fig. 1. The images we consider have ground-truth in the form of manual segmentation by humans. These manual segmentations are performed by several users independently. So, the number of clusters differs from user to user. The ground truth have only components information. A single cluster may have several components. Since this information is missing, we consider each component a single cluster. The manual segmentation is done mostly based on human perception, rather than on some computable features like color and texture. Martin et. al. [15] prepared this dataset and used it to evaluate the performance of segmentation algorithms and measuring probability distributions associated with Gestalt grouping factors as well as statistics of image region properties.

**Figure 1: Examples of images from the Berkeley segmentation database [15]. Image Ids are (from top left to right) "161062", "24063", "376020", "291000", "100075" and "113044".**

## 5.1 Color segmentation results

Let us consider the "161062" image. This image contains an identifiable pyramid object. The data cloud for this image in the hue saturation space is presented in Fig. 2(a). After applying the TASOCNN procedure, we obtain the processor graph as displayed in Fig. 2(b). Here the dots are the processors and edges are shown by black lines. This processor graph initially gives a total of 16 connected components, hence, 16 clusters. Before performing the TASOCNN clustering, we choose a threshold 0.0001 and the edges having weight less than this threshold are removed. Now, let us go through the TASOCNN clustering process. The inter-cluster edges are eliminated using the cost function defined in section 3. After this procedure many of the processors become isolated, so removed. We finally obtain a modified processor graph as shown in Fig. 3(a). Note, during the edge removal process the initial 16 clusters remains intact. The edge removal procedure can only increase the number of clusters. However, it generates several isolated processors. After removing isolated processors the number of clusters may be less than initial clusters. Here, we get a total of 18 clusters after edge removal process. In other words, we have 18 subgraphs in our modified processor graph.

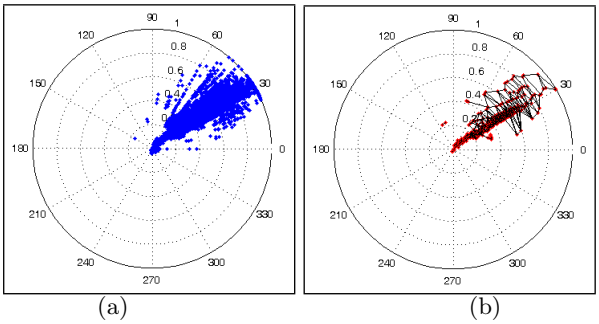



**Figure 2: (a) The data cloud of "161062" image in hue saturation space and (b) the initial processor graph for "161062" image.**

Now certain close subgraphs are to be merged. We set the distance threshold to be 0.4 that is found empirically. So any two subgraphs, at most 0.4 unit apart in spectral space, are considered to be close. We here have two different merging strategies. The single linkage principle takes the minimum distances among the processors to be the subgraph distance. The resulting set of graphs are shown in Fig. 3(b). The other one, which takes the maximum of the distance among processors to be the subgraph distance, results the processor graph as presented in Fig. 3(c). This procedure is similar to complete linkage principle. Here, the threshold 0.4 is used only with the complete linkage principle. Too few clusters (often one) are produced if we use the same threshold for single linkage principle. So, we make another experiment with the images and found 0.2 be a suitable distance threshold for single linkage based merging principle. We get 6

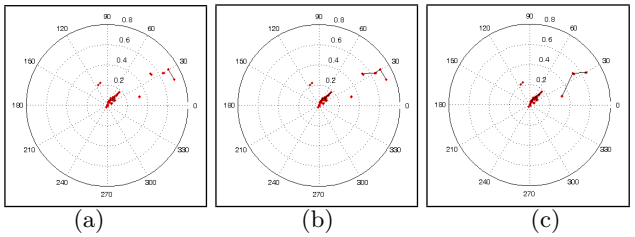



**Figure 3: (a) The processor graph after removing inter-cluster edges. (b) After performing single linkage based and (c) complete linkage based subgraph merging.**

clusters for single linkage based merging, whereas 4 clusters are produced by complete linkage based merging procedure. The results after performing subgraph merging are displayed in Fig. 4. The boundary of each connected component is marked in black. Here, we perform a $3 \times 3$ median filtering on the clustered images to remove certain perturbations. Observe, we have the boundary of the pyramid object, segmented into a separate cluster with the single linkage principle clustering. In contrast, it remains with the pyramid with complete linkage principle. In Fig. 5 we present some more results of color segmentation. In the images of Fig. 5, we may observe, several small connected components exists that are spatially close. These components belong to different spectral clusters. These components are produced since the spatial information is not considered during any stages of TASOCNN. Such spatially close components can be merged to form a large component by applying an appropriate merging procedure. However, we here we leave any spatial merging for future consideration.

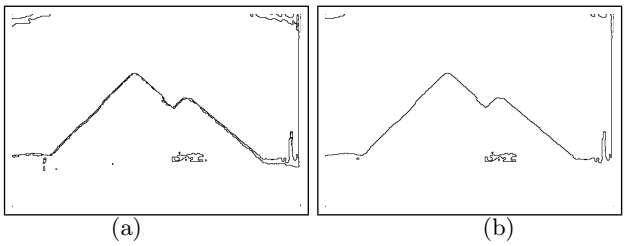



**Figure 4: The clustering results after performing (a) single linkage based and (b) complete linkage based TASOCNN clustering.**

## 5.2 Evaluation of segmentation and comparison

The evaluation is done with respect to the set of ground truth images. We again take the example image "161062".

**Table 1: Rand Indices for image Id. "161062" using (a) different TASOCNN clustering procedures and (b) using two other clustering algorithms.**

| Methods | User Id | | | | | PRI |
|---|---|---|---|---|---|---|
| | **1109** | **1115** | **1121** | **1123** | **1124** | |
| Single Linkage | 0.851 | 0.851 | 0.848 | 0.843 | 0.612 | 0.801 |
| Complete Linkage | 0.859 | 0.859 | 0.856 | 0.851 | 0.620 | 0.809 |

(a)

| Clustering methods | User Id | | | | | PRI |
|---|---|---|---|---|---|---|
| | **1109** | **1115** | **1121** | **1123** | **1124** | |
| K-means | 0.780 | 0.779 | 0.778 | 0.775 | 0.541 | 0.731 |
| vM-Gauss mixture | 0.714 | 0.714 | 0.711 | 0.706 | 0.470 | 0.663 |

(b)

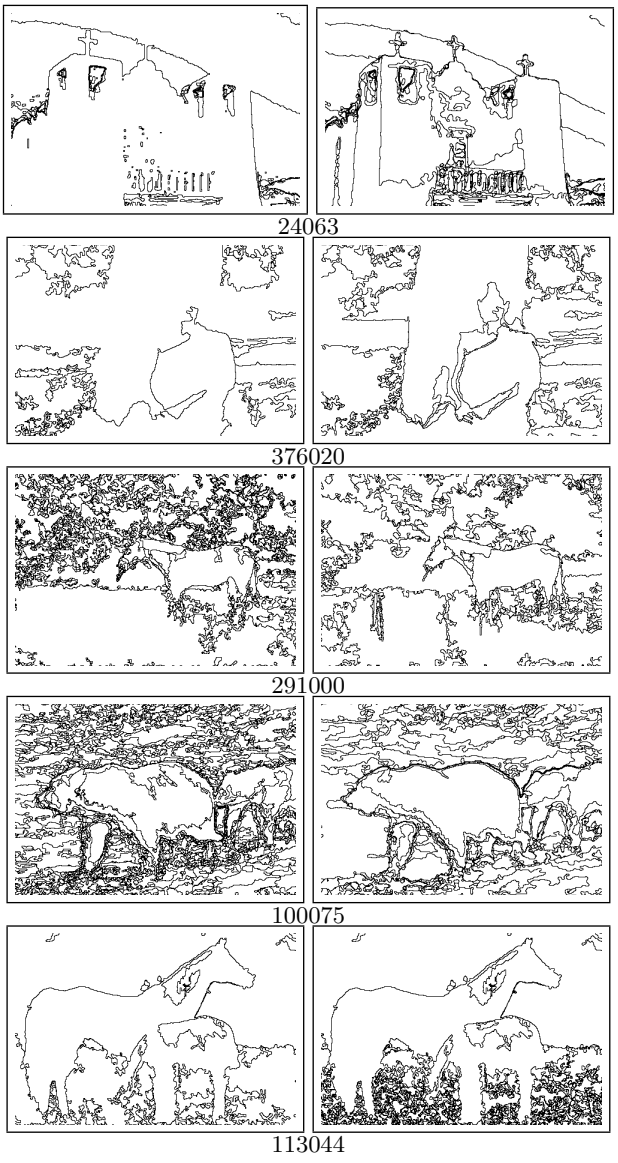



**Figure 5: TASOCNN based color segmentation results. Single linkage and complete linkage based clustering results are in the first and the second column respectively.**

Both the single linkage based and the complete linkage based results are evaluated by means of PRI. To analyze, let us observe the evaluation results for the five users, manually segmented the image. In table 1(a) we present the evaluation results for both type of TASOCNN clustering methods. Each cell in table 1 displays the rand index value for the concerned user. The last column presents the PRI, which can be expressed as an average of rand indices. To make a comparison, we present in table 1(b), the evaluation results after applying K-means and the vM-Gauss procedures both in the hue-saturation space. The K-means is performed using the same number of clusters as obtained from the complete linkage based TASOCNN clustering procedure (i.e. 4 in this case). The vM-Gauss mixture model was proposed by Roy et. al. [16]. Unlike K-means, this is a parametric procedure of clustering. vM-Gauss procedure assumes that hue-saturation information can be approximated by a mixture model of von-Mises and Gaussian joint distributions. The widely used Expectation Maximization algorithm is applied to estimate mixture parameters. The well-known Schwarz's Bayesian inference criterion (BIC) is used to predict the number of clusters.

According to table 1 we could note, the K-means gives better PRI than the vM-Gauss parametric clustering procedure. Also it gives better rand indices for all the users. However, the TASOCNN procedure seems to outperform both K-means and vM-Gauss mixture model. Before discussing further, we observe the performance results for some other images. In Fig. 6 we present the performance of complete linkage based TASOCNN clustering, the K-means using the same number of clusters as in TASOCNN and the vM-Gauss mixture model predicting number of clusters independently.

From Fig. 6 we could observe that K-means as well as the vM-Gauss model performs better than TASOCNN for the image Id. "113044". For other images, TASOCNN seems to outperform the vM-Gauss mixture model. To understand the relative performances of TASOCNN and vM-Gauss model, let us construct a synthetic dataset having two clusters. The data distribution of the two clusters are shown in Fig. 7. Here, one cluster (let $C_1$) is displayed in red while the other ($C_2$) in blue. The data in cluster $C_1$ are negatively correlated, whereas positive correlation is present among data in $C_2$. The vM-Gauss mixture model [16] assumes that hue and saturation are independent random variables. This may not be true always. When, highly correlated data are subjected to vM-Gauss mixture model, it may not correctly predict the number of clusters. As for our synthetic dataset, a total of 7 clusters are predicted according to the first local minima of the BIC criterion. The plot of BIC values is shown in Fig. 7(b). The TASOCNN,

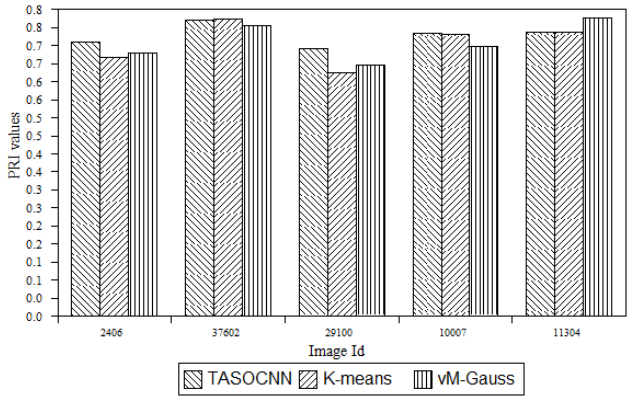
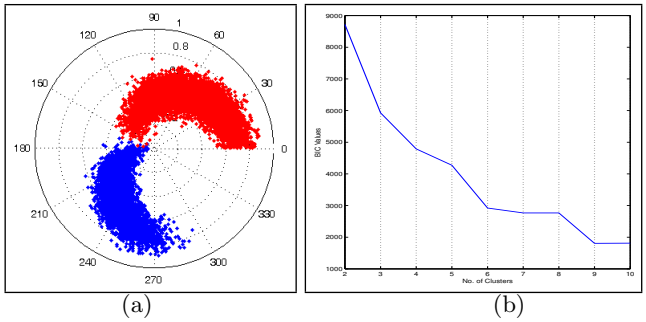
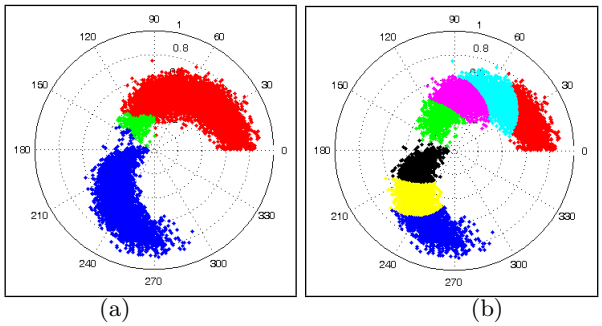



**Figure 6: Performance comparison of three algorithms with respect to PRI.**




(a) (b)

**Figure 7: (a) The Synthetic dataset consisting two clusters and (b) the BIC curve after applying vM-Gauss mixture model.**



(a) (b)

**Figure 8: (a) The clusters produced by TASOCNN and (b) vM-Gauss mixture model on the synthetic dataset.**

on the other hand, can estimate the number of clusters almost correctly and generating 3 clusters. The edges inside a cluster have enough strength. Some of the edges connect the processors situated in the two different clusters. These edges become weak and removed. The portion where $C_1$ comes in contact with $C_2$ have processors representing data in both clusters. This portion is separated into a single third cluster during TASOCNN clustering procedure. The clusters obtained by TASOCNN and vM-Gauss mixture model respectively are shown in Fig. 8(a) and Fig. 8(b). Even from the theoretical point of view, it is clear that TASOCNN can take care of correlation between hue and saturation while vM-Gauss mixture model cannot.

Continuing our experiments with Berkeley segmentation dataset, we observe, in most of the cases the TASOCNN outperforms the vM-Gauss procedure. The K-means algorithms, however, has a similar motivation as TASOCNN and can compete with TASOCNN in some cases.

## 6. SUMMERY AND FUTURE SCOPE

In this paper a variation of TASONN, namely, TASOCNN for circular-linear data is developed. The TASOCNN includes circular-linear updates and a circular-linear distance computation. A processor graph is constructed by connecting neighborhood processors of a data point. The different clusters of the data can be realized from the components of the processor graph. These components may be connected with inter-cluster edges. They are made disconnected by eliminating the inter-cluster edges by minimizing a cost function. The final set of connected components correspond to different clusters. The TASOCNN is applied on the problem of color based image segmentation. Some initial results are presented here. We compare TASOCNN with K-means and vM-Gauss mixture model. From the experiments done here, TASOCNN seems to outperform both K-means and vM-Gauss mixture based clustering. Though in our initial experiments we got satisfactory results, a close and elaborated study is needed in this direction. Issues concerning the TASOCNN include choice of learning parameters. Another critical issue is the choice of a suitable cost function. The current cost function may sometimes fail to produce a satisfactory result. From the results given in Fig. 5, it is evident that a good post-processing method may bring the results closer to the ground truth images. Concerning comparison, we should use more sophisticated and widely used procedures such as mean-shift or graph based clustering.

## 7. REFERENCES

[1] H. D. Cheng, X. H. Jiang, and J. W. Y. Sun. Color image segmentation: advances and prospects. *Pattern Recognition*, 34, 2259–2281, 2001.

[2] J. Lee, J. Wang, and C. Zhang. Color image segmentation: Kernel do the feature space. *Proceedings of the* $14^{th}$ *European Conference on Machine Learning*, 253–264. Springer, 2003.

[3] W. Tao, H. Jin and Y. Zhang. Color image segmentation based on mean shift and normalized cuts. *IEEE Trans. on SMC(B)* 37(5), 1382–1389, 2007.

[4] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, 2001.

[5] T. M. Martinetz and K. J. Schulten. Topology representing networks. neural networks. *Neural*

*Networks*, 7, 507–522, 1994.

[6] J. Vesanto and E. Alhoniemi. Clustering of the self-organizing map. *IEEE trans. on Neural Networks*, 11(3), 586–600, 2000.

[7] Y. Jiang, K. J. Chen, and Z. H. Zhou. Som based image segmentation. *Lecture Notes on Computer Sciences (LNCS)*, 2639, 630–643, 2003.

[8] A. Datta, S. K. Parui, and B. B. Chaudhuri. Skeletonization by a topology-adaptive self organizing neural network. *Pattern Recognition*, 34, 617–629, 2001.

[9] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(11), 1101–1113, 1993.

[10] D. R. Martin. *An Empirical Approach to Grouping and Segmentation*. PhD thesis, EECS Department, Univ. of California, Berkeley, 2002.

[11] R. Unnikrishnan, C. Pantofaru, and M. Hebert. Toward objective evaluation of image segmentation algorithms. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(6), 929–944, 2007.

[12] W. M. Rand. Objective criteria for the evaluation of clustering methods. *J. Am. Statistical Assoc.*, 66(336), 846–850, 1971.

[13] D. E. Ilea and P. F. Whelan. Ctex-an adaptive unsupervised segmentation algorithm based on color-texture coherence. *IEEE Trans. on Image Processing*, 17(10), 1926–1939, 2008.

[14] H. Zhang, J. E. Fritts, S. A. Goldman. Image segmentation evaluation: A survey of unsupervised methods. *Computer Vision and Image Understanding*, 110, 260–280, 2008.

[15] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *Proc. Int Conf. Computer Vision*, pages 416–423, IEEE Computer Soc., 2001.

[16] A. Roy, S. K. Parui, and U. Roy. A color based image segmentation and its application to text segmentation. In *Proc. of Indian Conference on Computer Vision, Graphics and Image Processing*, pages 313–319, IEEE Computer Soc., 2008.