Problem 1

1.1 Motivation

The texture content represents the spatial variation in pixel intensities and characterizes the regions in an image. The classification of objects can be realized based on different textures. One way is to extract the features from the images based on energy in different spectra and then use pattern recognition method to train classifiers.

1.2 Approach

1.2.1.

The **feature extraction** is to use Laws' filters to filter texture images and then calculates the local energy of the output images as features. The filter banks used in this problem are 25 Laws' filters. Each filter is a tensor product of 5 filters with 1D kernel. Since there are 25 filters, for each input texture image, there are 25 features (i.e. a sample has 25 features). The details about feature extraction is discussed in (1.3 a-1).

1.2.2

The classification used here is **Minimum Mean Distance classifier** based Mahalanobis distance instead of Euclidian distance. Because Mahalanobis distance is a relative distance compare with Euclidian distance, it can be more accurate. If a sample has the smallest Mahalanobis distance to one class mean, then it belongs to this class (1.3 a-2).

1.2.3

One feature dimension reduction is **PCA**. The PCA transform all samples (ignore the class labels) in the feature space into a new feature space (the axes corresponding to the ordered eigenvalues). So in this new feature space, we can select the fewer axes (features) to represent the samples according to the eigenvalues (1.3 a-3). The code used is function *pca* from MATLAB.

1.2.4

Another feature dimension reduction is **LDA**. LDA transform the sample into (C-1) dimension feature space based on class labels. The idea is try best to classify the samples in this new feature space (1.3 a-4). The code used is function LDA from online source.

1.2.5

Besides Minimum Mean Distance classifier, another classifier used in part (b) is **Support Vector Machine** (SVM). SVM can build a model and use this model to assign class labels to input samples (1.3 b-3). In this problem, I use the LIBSVM [1]. LIBSVM implements the SMO algorithm for kernelized support vector machines.

1.3 Results & Discussion

a-1

There are 96 texture images so the number of samples are 96. Pre-processing is to extract local mean form the original pixel value to remove the dc component. I use a 5x5 filter centered at each pixel to calculate the local mean. And then subtract it from its original intensity value. There are two 1D kernel filters provided, to from 25 Laws' filter banks, I take tensor product of any two of them (two 1D filters can be the same). Then apply the 2D filter to the image after pre-processing. Each pixel of the output image stands for energy and since there is only one type of texture, I take average energy of 128x128 pixels, the mean represents one feature. So for an image, 25 filters result in 25 features, these comprise the feature space. After applying the process to 96 texture images, the dataset for part(a) is 96 samples with 25 features and for part(b). Each sample can be denoted as a vector $\vec{x}_{1\times 25}$.

a-2

Mahalanobis distance, $D_M(\vec{x}) = \sqrt{(\vec{x}-\vec{u})^T\Sigma^{-1}(\vec{x}-\vec{u})}$. Σ stands for the covariance matrix for training samples belonging to same class. $\vec{u}$ stands for the mean for each class. In this question, the dimension of feature is 25 and there are two classes. First of all, I calculate the Σ and $\vec{u}$ for grass and straw respectively according to the training samples (36 grass samples and 36 straw samples). Then assign class label to the unknown sample (24 test samples) based on comparing the Mahalanobis distances to different class means.

The error rate for training and test data is shown in first row in the Fig.1

a-3

For PCA method, the **25 features are not equally important**. Since different features have different eigenvalues, they have different effect on classification. The basic idea for PCA is that in each feature domain, the samples have different distribution (variance). The larger variance the samples have, the better the samples will be classified. So PCA calculates the eigenvalues and corresponding eigenvectors for feature space and rearranges the features in descending order. Then use the eigenvectors to transform the original feature space to a new feature space (the number of eigenvectors depending on dimension we expect), in this case, we want to reduce the dimension from 25 to 1, so the transform matrix has one eigenvector with the largest eigenvalue.

The error rate for training and test data is shown in second row in the Fig.1

a-4

LDA takes the labeled samples into account which is different for PCA. The basic idea for LDA is that after transforming the original feature space to a new feature space, each class has the smallest variance and the largest distances between each class mean. In this situation, the samples will have the best classification result. LDA can reduce the dimension of feature space to (C-1), where C stands for the number of class. As stated in PCA, the LDA method finds a transform matrix whose column is C-1. In this case, there are only two classes, grass and straw, so the new feature space has 1 dimension.

The error rate for training and test data is shown in third row in the Fig.1

a-5

	training data	test data
Original feature space (25 features)	0%	0%
PCA (1 feature)	0%	0%
LDA (1 feature)	0%	0%

Figure.1 the error rate for training and test data of different classifiers

a-6

From Fig.1, all the three methods can correctly classify the unknown samples. Compared with PCA and LDA, using the original feature space is more complicated because it involves 25 dimensions and the computation is really huge. So PCA and LDA are better than the first one.

As for PCA and LDA, the new feature space in Fig.2 shows that LDA gives a relatively larger distance between the two class means than PCA does. In other words, LDA classifier leads to a better generalization to unknown data (aside the 24 unknown data). So LDA performs better than PCA.

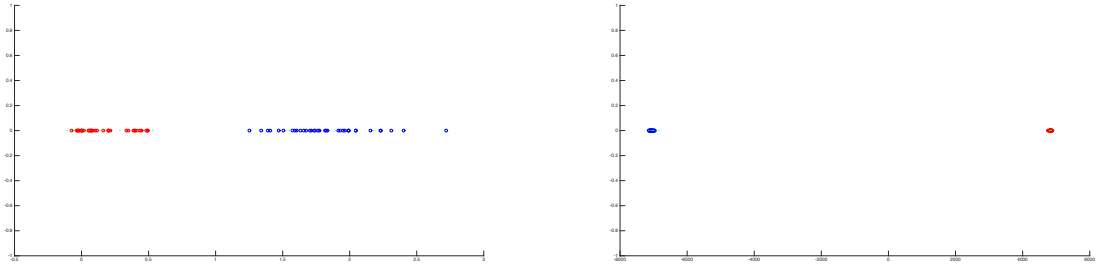


Figure.2 the new feature space for PCA and LDA
(red ‘o’ is grass sample, blue ‘o’ is straw, green ‘·’is the unknown sample (100% correctly classified))

b-1

There are four classes in feature space, grass, feature, sand and straw. The classifiers used here are binary classifiers. PCA method can reduce the feature space dimension to any number while LDA reduces the feature space dimension to C-1. But in this case, we use binary classifiers, the number of class is two, positive (class

of interest) and negative (class of the other three classes). So LDA reduces the dimension to 1. In this problem, I use PCA to reduce the dimension to 1 and 3 respectively and use LDA to reduce the dimension to 1.

The error rate for training and test data is shown in the Fig.3. Since the training and test samples are selected randomly, I run the classifier **20 times** to get the average error rate for each classifier.

b-2

PCA (1-D)	Training data	Test data
Grass vs no.G	0.349	0.392
Feature vs no.F	0.119	0.129
Sand vs no.S	0.206	0.236
Straw vs no.S	0.457	0.463

PCA (3-D)	Training data	Test data
Grass vs no.G	0.029	0.042
Feature vs no.F	0.016	0.025
Sand vs no.S	0.001	0.004
Straw vs no.S	0.134	0.177

LDA (1-D)	Training data	Test data
Grass vs no.G	0.083	0.094
Feature vs no.F	0.068	0.077
Sand vs no.S	0.029	0.053
Straw vs no.S	0.150	0.258

Figure.3 the error rate for training and test data for PCA and LDA

b-3

In original feature space, the boundaries between samples in different classes may be nonlinear. SVM uses a kernel to map the original samples to φ space. In the new space, the boundaries are linear and SVM try to find the gap between different classes as wide as possible. Then map the boundaries back to the original space. In this problem, I use libSVM form MATLAB, the first step is to use training samples to train the model (function *svmtrain*). Then use the trained model to predict the labels for the samples (function *svmpredict*).

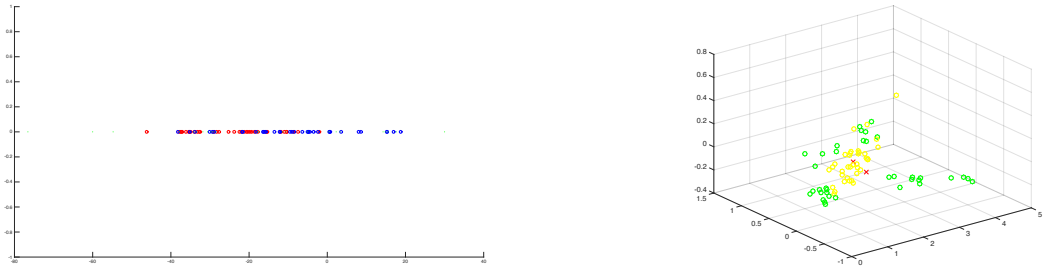
The error rate for training and test data is shown in the Fig.4.

SVM	Training data	Test data
Grass vs no.G	0.008	0.010
Feature vs no.F	0.002	0.003
Sand vs no.S	0.005	0.007
Straw vs no.S	0.013	0.018

Figure.4 the error rate for training and test data for SVM

b-4

According to Fig.3 and Fig.4, the best classifiers are binary classifiers using SVM method. When the feature space dimension is 1, the PCA results in a poor error rate. Because PCA doesn't consider the class label and 1-D provides not enough information (under-fitting) to classify the samples. Compared LDA with PCA using 3 dimension features, even if LDA consider the class label, LDA classifier is under-fitting so the performance is worse than PCA using 3 dimension. Fig.5 (a) shows that in LDA in 1-D case, the two class samples overlap in some extent, so the unknown samples in this area can be easily misclassified based on the smallest distance. (b) shows that there is also a problem in PCA (3-D) case, but in 3-D feature space, the situation is better. Fig.5 shows that it's not good to use minimum mean distance classification, applying SVM is a better choice. Since SVM finds a better boundary between the two classes as described in (b-3). So SVM avoids the overlapping and under-fitting issue.



(a) LDA (1-D) (b) PCA (3-D)

Figure.5 the new feature space for LDA and PCA (3-D)
(in (a), red 'o' stands for positive training sample, blue 'o' stands for negative training sample, green '·' stands for test samples. in (b) yellow 'o' stands for positive training samples and green 'o' stands for negative samples, red 'x' is the mean for each class)

Problem 2

2.1 Motivation

In digital image, edges can be described as curved lines composed of the points where brightness change sharply. They are the boundaries of the objects and can be used to capture important information in image. Some methods about edge detection and their performance evaluation are discussed in this part.

2.2 Approach

2.2.1

RGB to grayscale: The original formula is Gray = R*0.299 + G*0.587 + B*0.114. To improve the calculation speed, I use shifting operation, Gray = (R*38 + G*75 + B*15) >>7. Since it is more accurate and faster than floating operation.

2.2.2

Sobel edge detector is 1st order derivatives. The way to get the gradient magnitude is to calculate the derivate for each pixel in x direction and y direction. The formula is $\frac{\partial F}{\partial x} = \frac{1}{4}(A_2 + 2A_3 + A_4 - A_0 - 2A_7 - A_6)$ and $\frac{\partial F}{\partial y} = \frac{1}{4}(A_0 + 2A_1 + A_2 - A_6 - 2A_5 - A_4)$. A stands for the neighboring pixel value. Then the magnitude is $\sqrt{\frac{\partial F^2}{\partial x} + \frac{\partial F^2}{\partial y}}$. After getting the magnitude for each pixels, I picked 10% or 15% of them with the largest magnitude value as the edge points and the rest pixels are background.

2.2.3

Non-maximal suppression: For this case, I use a simpler non-maximal suppression to decide whether the edge points should be preserved or not by comparing the target gradient magnitude with the neighboring up and down pixels or right and left pixels. For each edge point from Sobel edge detector, I calculate the direction of the gradient, $\theta = \tan^{-1}(\frac{\partial F}{\partial y}/\frac{\partial F}{\partial x})$. If $\theta > \frac{\pi}{4}$, I compare the target gradient magnitude with neighboring up and down pixels and vice versa. I the target gradient magnitude is the largest, then this edge point is preserved.

2.2.4

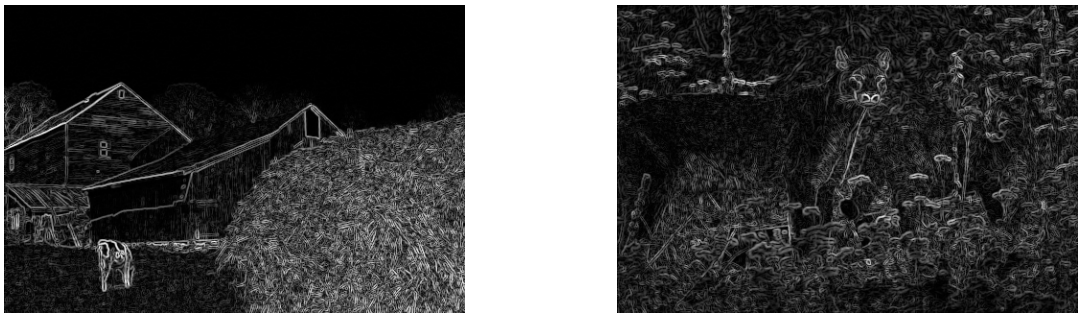
Canny detector: Canny detector integrated a multi-stage algorithm to detect the edges. In this problem, I use function *Canny* from opencv to get the edge map. There are three parameters needed to be set. First two are low threshold and high threshold (2.3 b-1). The last parameter is aperture size, I set it 3.

2.2.5

Structured Edge detection (SE): SE method is a new edge detection method in recent years, and the details are discussed in (2.3 c-1)

2.3 Results & Discussion

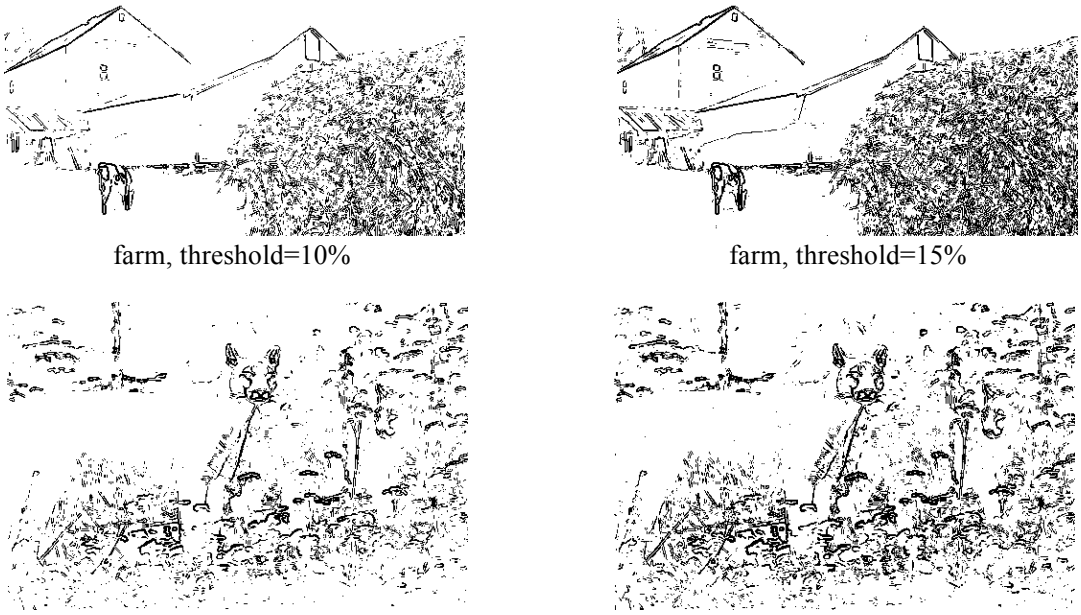
a-1



(a)Farm (b)Cougar

Figure.6 Normalized gradient magnitude map for Farm and Cougar images by Sobel edge detector

a-2

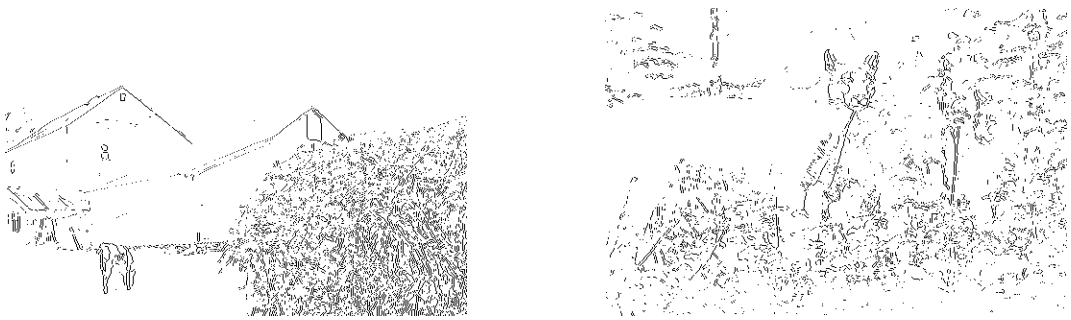


farm, threshold=10% farm, threshold=15%

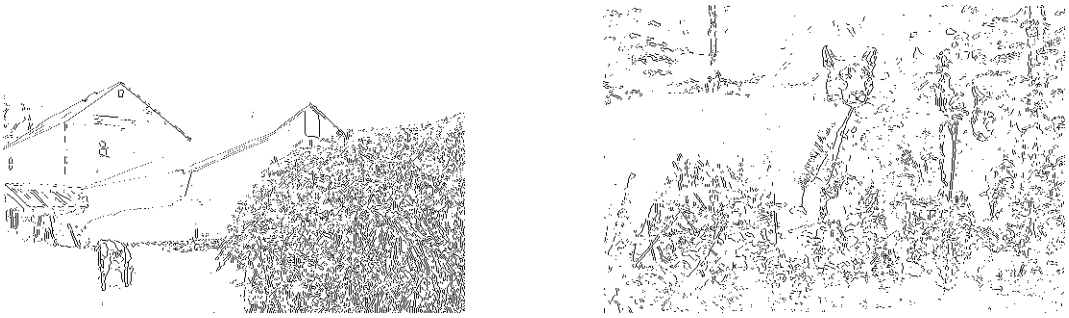
cougar, threshold=10% cougar, threshold=15%

Figure.7 Edge maps for Farm and Cougar images with different thresholds by Sobel edge detector)

a-3



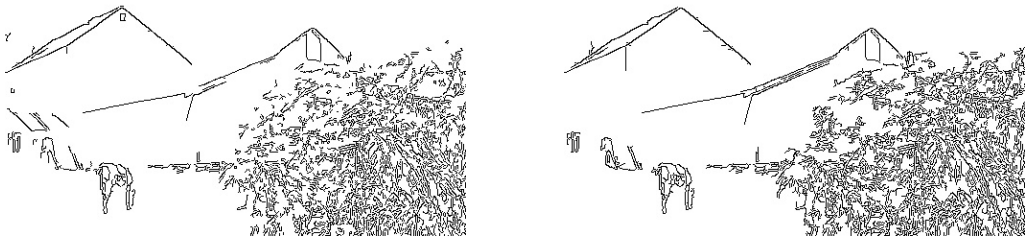
farm, threshold=10% cougar, threshold=15%



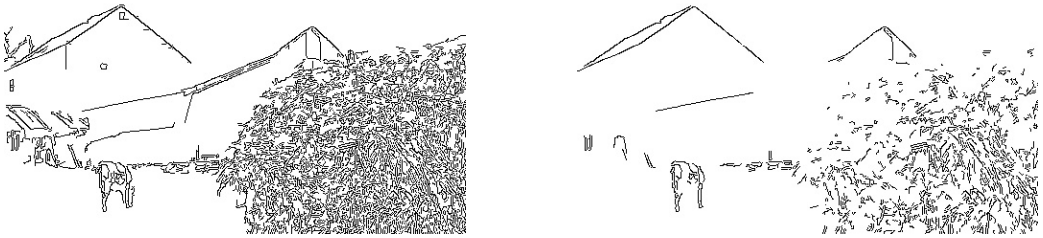
farm, threshold=10% cougar, threshold=15%

Figure.8 Enhanced edge maps for Farm and Cougar images by non-maximal-suppression

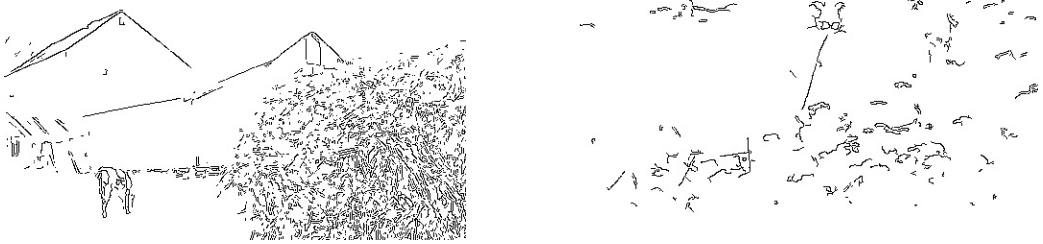
b-1



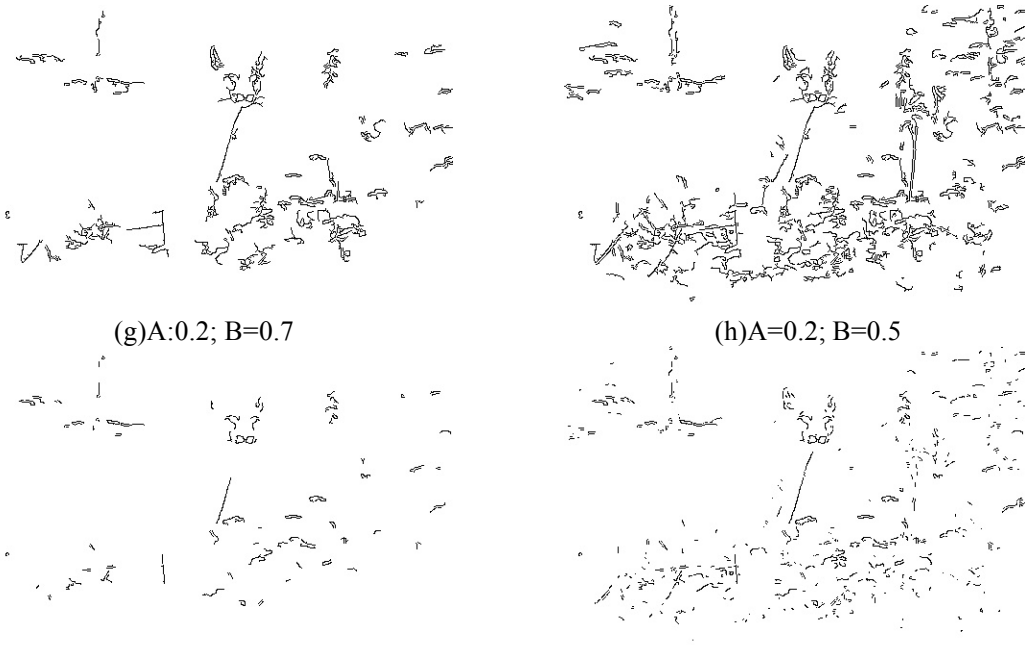
(a) A: 0.3; B: 0.6 (b)A: 0.2; B: 0.7



(c)A: 0.2; B: 0.5 (d)A: 0.4; B: 0.7



(e)A: 0.4; B: 0.5 (f)A: 0.3; B: 0.6



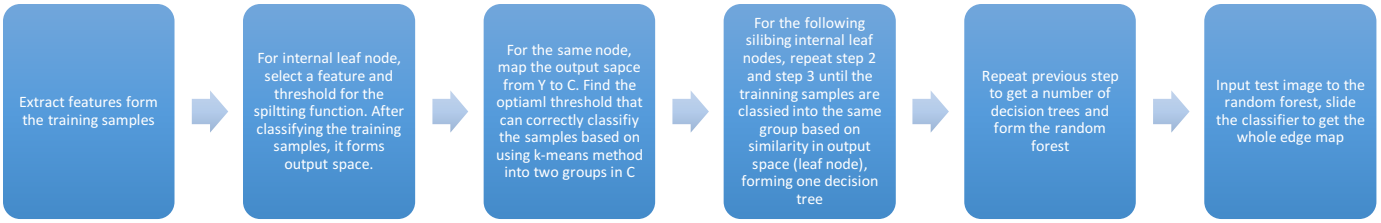
(g)A:0.2; B=0.7 (h)A=0.2; B=0.5

(i)A: 0.4; B: 0.7 (j)A: 0.4; B: 0.5

Figure.9 Edge maps for Farm and Cougar images using different low and high thresholds by Canny edge detector

The **high threshold** is used to find the **edges against the background**. When the high threshold is small, it results in more boundaries and edges, compared (b) with (c), (d) with (e), (g) with (h), (i) with (g), where the low threshold is the same and high threshold varies. When B is lower, there are more edges. The region above the high threshold is strong object boundary regions. If the high threshold is very big, the edges can be discontinuous and not smooth, so low threshold is used. The **low threshold** is used for determining the pixels below this value as the **background** and the region between low and high threshold is the **weak object boundary region**. It can add some edges to the edges determined by the high threshold, making the boundary smoother and resulting more partitions, compared (b) with (d), (c) with (e), (g) with (i), (h) with (j), where the high threshold is the same, but the low threshold varies. When A is lower, the weak object boundary region is wider and can add more edges. In the weak object boundary region, the points near the edge points tend to be determined as edge points to make the boundary or edge continuous. If there are no edge points around, the points tend to be determined as background points.

c-1



SE algorithm uses input patches of image to predict local segmentation masks and then uses these structured labels to train decision trees, forming random forest. For patches of the input test image, random forest gives edge pixel labels which are assembled and computed to give the final edges.

Step1: We have a enough number of image patches (some with different kinds of edges, some with on edges) as the training samples. The approach assumes the input features have two types: **pixel lookups** and **pairwise pixel differences**. To find the pixel lookups, it predicts a structured 16x16 segmentation mask from 32x32 image patch, considering different channels with multiple additional information that indicate whether a pixel is an

edge or not [6] [7]. The total number of channels is 13, including 3 color channels, 2 magnitude and 8 orientation channels. All channels are down-sampled by 4, so the total pixel lookups features are 32*32*13/4=3328. The pairwise difference features are form all candidate pairs from 5*5 pixels per channel, 13*C_{25}^{2}=3900. So the total number of features is 7228. When building the decision tree, **not all the features are used**. Because we need to avoid the decision being too deep, causing over-fitting problem. And actually, after some hierarchies (not too large), the decision tree is well trained.

Step2: To build the decision tree, we need to select the feature from step1 and threshold for splitting function for each internal leaf node. The training samples through the node form the output space. In this output space, each sample has a specific type of edge or no edge. ($y \in Y$ stands for corresponding structured labels for each input image patch x ($x \in X$)).

Step3: Since Y is structured labels from 16x16 segmentation masks, using binary to indicate the outcomes, there will be 2^{16*16} (dimensionality). This output space Y has high dimensionality, it's expensive to compute. Also, it fails to represent the information of the image. For instance, some outcomes are same. Based on this, the information can be preserved in lower dimensionality space. So the approach utilizes two mapping: Y→Z and Z→C.

Form Y to Z, since the information relies on the similarity over Y, so each pixel has no information about structure, but the pairwise pixels carry the information. So the dimensionality can be reduced to C_{16*16}^{2}. This approach sample m dimensions (m=256) of C_{16*16}^{2}, because it can lead to quicker computation and avoid randomness for training. The sampling ensures the sufficient diversity of tress.

From Z to C, using PCA to project the 256 dimensional space to at most 5 dimensional space. In the 5 dimensional space, using the k-means method (k=2) to train each internal leaf node of the tree to correctly divide the samples entering the internal leaf node the into two classes. After this, we can decide whether the feature we select for the internal leaf node is available or not and find the optimal threshold for the splitting function.

Step4: Iterating the previous steps, we will form a decision tree. In leaf node, the samples will have the sample type of edge (structured labels). For the input test patch, it will finally enter one of the leaf nodes and we will get the structured label.

Step5: Build a number of different decision trees to form random forest. For the input test patch, the decision trees may result in different structured labels. We make our final decision on the occurrence of different structured labels.

Step6: Once we trained the random forest, we can slide the random forest on the input test image row by row to get the edge map.

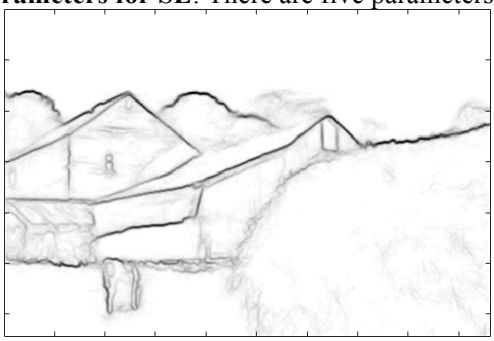
c-2

Decision construction: A decision tree has two kinds of nodes, leaf node and internal leaf node. It is a flow-chart-like structure. Internal leaf node has a spilt function, including feature and threshold. For the input training sample, the internal leaf node will classify the it into different following nodes. Iterate this many times and the sample will reach the leaf node finally. Leaf node set a label to the samples reaching it. The main task is to chose optimal features and thresholds for the internal leaf node. It is based on the mutual information after training samples being classified by this node. To avoid over-fitting, the decision tree should not be grown too deep.

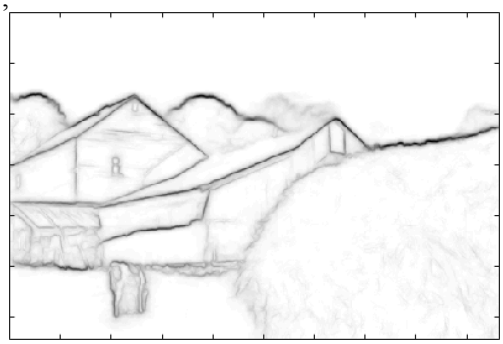
Principle of RF classifier: The random forest classifier is an ensemble method, constructing a number of decision trees to classify samples. It can improve the classification rate. To achieve this, the principle is that since each specific decision tree we get, has a very low classification rate, it may over-fit for new data. So after we ensemble them, due to the law of large numbers, the RF classifier can have a much better classification rate, it can avoid over-fitting problem to new data set.

c-3

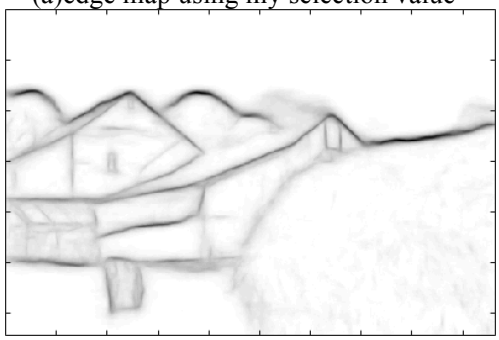
Parameters for SE: There are five parameters to be consider,



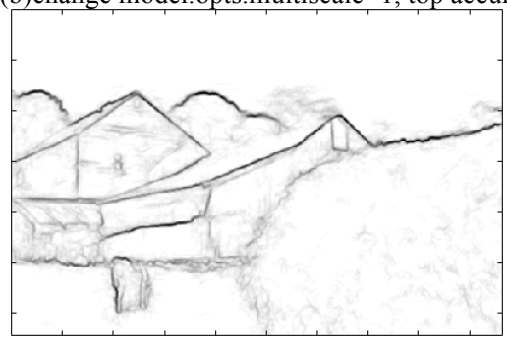
(a)edge map using my selection value



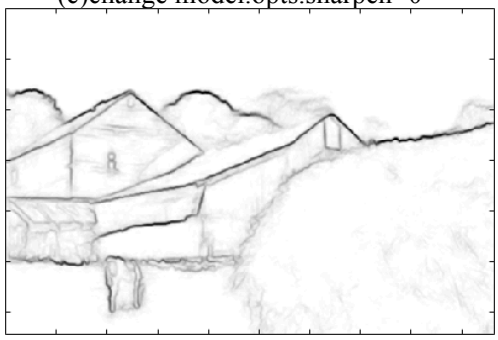
(b)change model.opts.multiscale=1, top accuracy



(c)change model.opts.sharpen=0



(d)change model.opts.nTreesEval=1



(e)change model.opts.nThreads=1

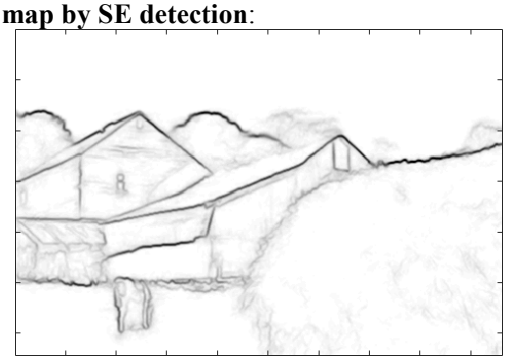
Figure.10 Edge map by SE detector with different parameters (use farm image as example)

From Fig.10, compared with (a), (b) has more accuracy although it is not evident for farm image. (c) is more blurring, because this parameter determines the sharpness of edges. When the value increases, the edges become sharper, (d) can reduce the running time, the running time for (a) is about 0.2s while (d) is about 0.06s. (e) determines the threads for evaluation, since the pixel values of edge map we get are not just 0 and 1, we need to set threads to determine 0 and 1. This parameter seems have no effect on the output edge map.

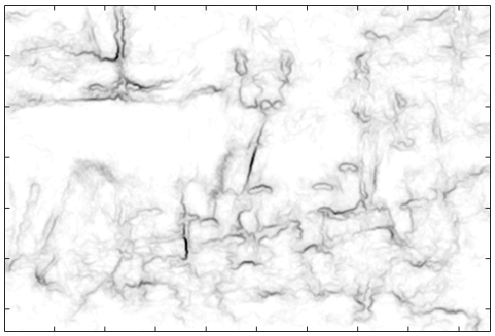
Final conclusion:

model.opts.multiscale	Determine the accuracy of the edge map, here I use 0
model.opts.sharpen	Determine the sharpness of the edge map, here I use 2
model.opts.nTreesEval	Determine the number of trees for evaluation, affect the running time, here I use 4
model.opts.nThreads	Set the maximum threshold for evaluation, here I use 4
model.opts.nms	Determine whether to use NMS or not, here I use 0, not using NMS

Edge map by SE detection:



(a)farm



(b)cougar

Figure.11 Edge maps for Farm and Cougar (Structured Edge)

Fig.7 shows that the edges by **Sobel edge detection** are wider. It has more redundant edge points (can be regarded as false positive points). Also there are lots of discontinuous edge points, which are useless. Fig.9 shows that **Canny edge detector** improves the visual result compared with Sobel edge detector. It reduces lots of redundant and useless edge points, making the edge narrower. But it loses some important edge information, for instance, the figure of cougar is incomplete. Also, it has some unimportant edge points. For example, the mow in farm has lots of details edges. **SE detector** overcome the shortcoming of Sobel edge detector and Canny edge detector. It has thin edges of objects in image and also preserve the complete edge information.

(d)-1

For each groundtruth, I found the largest F-measure, use the corresponding precision and recall for this groundtruth. After finding the five precision and recall, I calculate the average value to get the F-measure.

sobel detector (threshold 10%)							
farm	precision	recall	F-measure	cougar	precision	recall	F-measure
gt1	0.065	0.479		gt1	0.325	0.723	
gt2	0.123	0.544		gt2	0.307	0.745	
gt3	0.129	0.539		gt3	0.109	0.662	
gt4	0.120	0.649		gt4	0.344	0.764	
gt5	0.122	0.565		gt5	0.211	0.733	
mean	0.112	0.555	0.186	mean	0.259	0.725	0.382

sobel detector (threshold 15%)							
farm	precision	recall	F-measure	cougar	precision	recall	F-measure
gt1	0.055	0.566		gt1	0.268	0.837	
gt2	0.109	0.677		gt2	0.258	0.880	
gt3	0.118	0.687		gt3	0.094	0.804	
gt4	0.105	0.797		gt4	0.275	0.859	
gt5	0.110	0.709		gt5	0.172	0.840	
mean	0.099	0.687	0.174	mean	0.213	0.844	0.340

NMS (threshold 10%)							
farm	precision	recall	F-measure	cougar	precision	recall	F-measure
gt1	0.069	0.435		gt1	0.358	0.667	
gt2	0.113	0.433		gt2	0.340	0.691	
gt3	0.122	0.437		gt3	0.124	0.627	
gt4	0.114	0.534		gt4	0.384	0.714	
gt5	0.112	0.446		gt5	0.238	0.694	
mean	0.106	0.457	0.172	mean	0.289	0.679	0.405

NMS (threshold 15%)							
farm	precision	recall	F-measure	cougar	precision	recall	F-measure
gt1	0.062	0.522		gt1	0.303	0.798	
gt2	0.111	0.565		gt2	0.293	0.843	
gt3	0.121	0.576		gt3	0.108	0.779	
gt4	0.110	0.681		gt4	0.315	0.832	
gt5	0.111	0.590		gt5	0.197	0.812	
mean	0.103	0.587	0.175	mean	0.243	0.813	0.374

canny detector A:0.3 B:0.6							
farm	precision	recall	F-measure	cougar	precision	recall	F-measure
gt1	0.060	0.419		gt1	0.473	0.296	
gt2	0.115	0.482		gt2	0.440	0.300	
gt3	0.120	0.473		gt3	0.178	0.303	
gt4	0.113	0.576		gt4	0.542	0.339	
gt5	0.114	0.497		gt5	0.355	0.348	
mean	0.105	0.489	0.172	mean	0.397	0.317	0.353

canny detector A:0.2 B:0.7							
farm	precision	recall	F-measure	cougar	precision	recall	F-measure
gt1	0.058	0.464		gt1	0.413	0.304	
gt2	0.102	0.492		gt2	0.384	0.308	
gt3	0.109	0.495		gt3	0.163	0.327	
gt4	0.100	0.594		gt4	0.469	0.345	
gt5	0.102	0.515		gt5	0.328	0.378	
mean	0.094	0.512	0.159	mean	0.351	0.332	0.342

canny detector A:0.2 B:0.5							
farm	precision	recall	F-measure	cougar	precision	recall	F-measure
gt1	0.058	0.571		gt1	0.349	0.583	
gt2	0.105	0.631		gt2	0.315	0.573	
gt3	0.116	0.654		gt3	0.109	0.494	
gt4	0.103	0.754		gt4	0.371	0.620	
gt5	0.106	0.665		gt5	0.233	0.608	
mean	0.098	0.655	0.170	mean	0.275	0.576	0.373

canny detector A:0.4 B:0.7							
farm	precision	recall	F-measure	cougar	precision	recall	F-measure
gt1	0.080	0.325		gt1	0.561	0.140	
gt2	0.143	0.348		gt2	0.452	0.123	
gt3	0.145	0.334		gt3	0.216	0.147	
gt4	0.133	0.398		gt4	0.613	0.153	
gt5	0.140	0.357		gt5	0.384	0.150	
mean	0.128	0.352	0.188	mean	0.445	0.143	0.216

SE detector							
farm	precision	recall	F-measure	cougar	precision	recall	F-measure
gt1	0.408	0.808		gt1	0.478	0.595	
gt2	0.684	0.819		gt2	0.453	0.617	
gt3	0.756	0.809		gt3	0.242	0.503	
gt4	0.624	0.830		gt4	0.459	0.678	
gt5	0.680	0.849		gt5	0.393	0.561	
mean	0.630	0.823	0.714	mean	0.405	0.591	0.480

Figure.12 F-measure for Edge maps using different methods

From Fig.12, the performances of Sobel detector, NMS and Canny detector are worse than the SE detector. For **Sobel detector**, as shown in Fig.7, the edges are very wide and there are lots of isolated edge points, accounting for lots of false positive points. So the precision is very low. After improved by the **non maximum suppression**, some redundant and noisy points are removed, so the precision becomes larger. Since most pixels are determined as edge points, so the number of false negative points is small, resulting in a higher recall. Totally, the Sobel detector has a lower precision but higher recall because it gets high recall at cost of loss of precision. When using **Canny detector**, if the high threshold is low, most pixels are determined as edge points, accounting for higher recall but lower precision and vice versa. So we can adjust the high and low threshold to obtain an optimal performance of edge map. For **SE detector**, it can get a better precision and recall at the same time, reducing the number of false positive and false negative points. Because it trains a model to detect the edges in an image.

(d)-2

Farm image is easier to get higher F measure. The farm image has simpler edge structure and the objects are monotonous while for the cougar image, the edges of the cougars and the background (tussock) are more complicated. For example, the tussock is a good camouflage for the cougars, even for human, it's hard to notice them. In farm image, the background is the sky and grass, the farmhouse, cow and hay mow are very easy to distinguish. Because of this, the number of false positive and negative points increases for cougar image.

(d)-3

Since $F = 2\times\frac{P\times R}{P+R} = \frac{2}{\frac{1}{R}+\frac{1}{P}}$, when P is significantly higher than R, F can be writhen as 2xR, it's not possible to get a high F and vice versa. When the sum of P and R is a constant, assume P+R=C, R=C-P, then $F = 2\times\frac{P\times R}{P+R} = 2\times\frac{P\times(C-P)}{P+C-P} = 2\times\frac{P\times(C-P)}{C}$. To get the highest F, P=$\underset{P}{\text{argmax}}\, \text{P}\times(\text{C}-\text{P}) = \frac{C}{2}$, so P=R=$\frac{C}{2}$.

Problem 3

3.1 Motivation

A digital image can be partitioned into several segments, making the objects or scene in the image more understanding and easier to be analyzed. There are three levels for image segmentation. The low level study the homogeneity of pixels in image, such as superpixel. The middle level is about larger segmentations, two methods in this level, Mean-shift method and Contour-guided Color Pallette (CCP) are discussed in this problem. The high level is semantic about object detection.

3.2 Approach

3.2.1

Sean-shift clustering: Starting at one seed point, define a radius parameter and find the mean in the neighborhood. Then move the starting point to the mean of this region. Iterate this process until the seed point can't move any more.

3.2.2

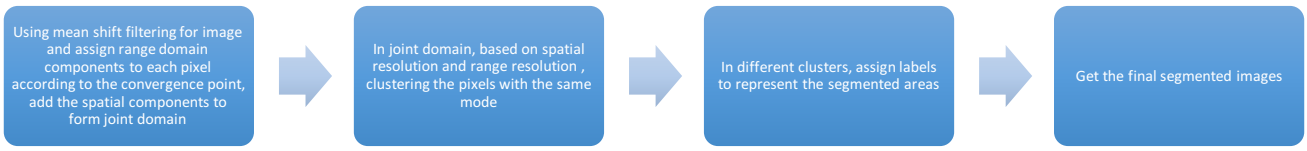
Mean-Shift (MS) Segmentation. The MS algorithm is described in (3.3a-1). *The code is online source code.*

3.2.3

Contour-guided Color Pallette (CCP) applies a bilateral filter for the original image in LAB color space. Use SE to get the edge map for the original image. Then pick color samples along the long contour to form the color pallette. Quantize the original image based on this color pallette. Finally do the post-processing, including the leakage avoidance by contours, fake boundary removal, small region mergence [3].

3.3 Results & Discussion

a-1



In the first step, using mean-shift clustering (3.2.1) to find the convergence points for each pixel in the original image. Then assign the range components to each pixel according to its convergence pixel range value. After this process, the pixels in filtered image have spatial and range components, forming the joint spatial-range domain.

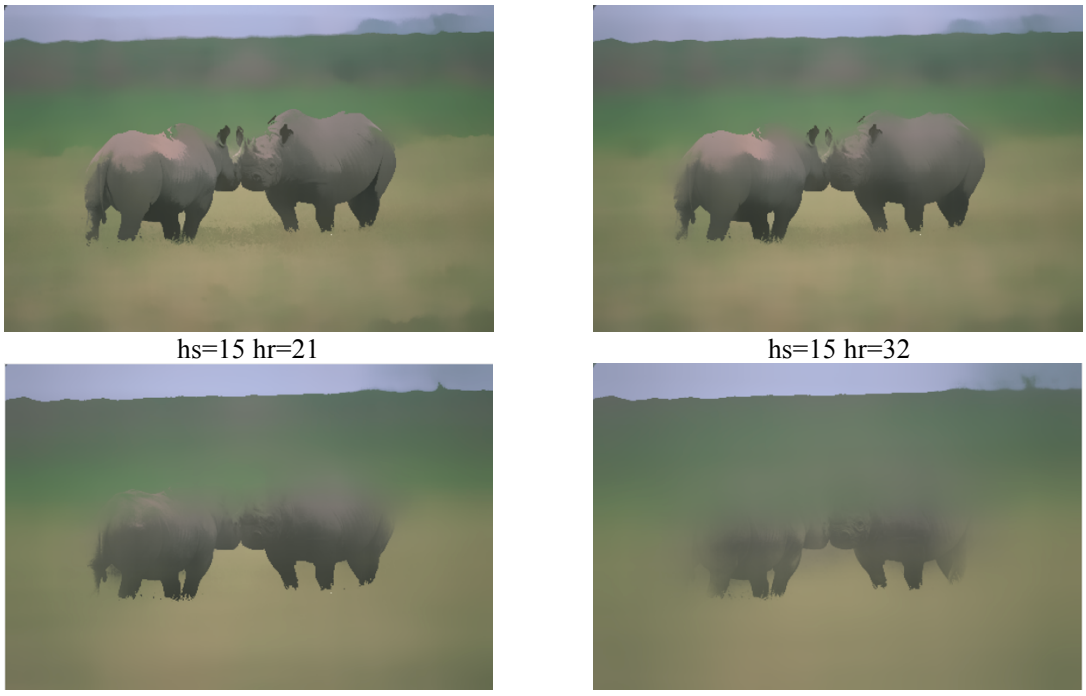
The second step is to use a spatial filter (bandwidth hs) and a range filter (bandwidth hr) to group the pixels into the same cluster in the spatial-range domain. Each pixel is put into one cluster.

The third step is to label each cluster, denoting different segmented areas. The last step is to assign color depending on the cluster label. [2]

The **key parameters** in implementing the MS algorithm is the **resolution** in spatial and range domain (i.e. the bandwidth of the kernel). In the following the Fig.13, 'hs' stands for the bandwidth of spatial kernel and 'hr' stands for the bandwidth of range kernel.

(a)-2

hs=15 hr=21 hs=15 hr=32 hs=32 hr=42 hs=50 hr=60



hs=15 hr=21 hs=15 hr=32

hs=32 hr=42 hs=50 hr=60

Figure.13 effect on the segmentation results with different resolution

In Fig.13, as the spatial bandwidth and range bandwidth increase, the **resolution of the segmentation image decreases** and **the segmented regions become less** (more and more parts become the same). When the **spatial bandwidth** increases, some regions in the image mix up, because pixels in different segments are put in the same cluster. When the **range bandwidth** increases, it results in some color deviation. The number of clusters decreases.

When the bandwidth is small, the segmentation effect is not very good, lots of unimportant details are preserved. For instance, when hs=15 and hr=21 in the Man image, the man's nose and the details of his clothes can still be distinguished. When bandwidth is very large, the important information is lost. For example, when hs=50 hr=60 in Rhinos image, some parts of Rhinos are merged into the background (the grass). Then we can hardly tell the rhinos in the picture.

b-1

original Man image denoised Man image

original Rhinos image

denoised Rhinos image

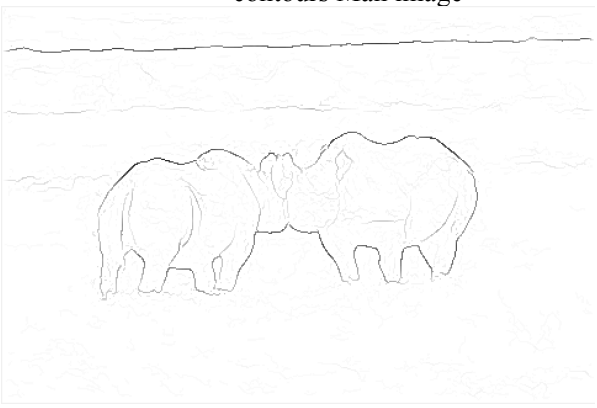
Figure.14 denoised Man and Rhinos images

Compared with the original images, the denoised images are more smoothing. The purpose of this step is to **remove the noise**, and **preserve the similarity in pixel values**. **Reduce the number of disturbing color in the range domain.**

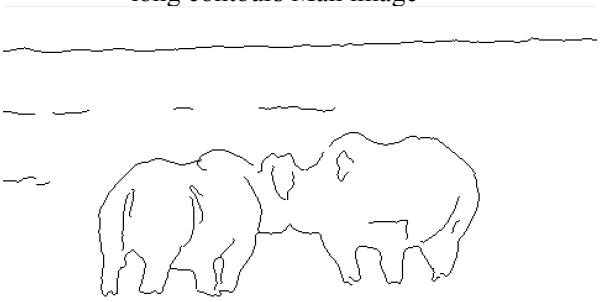
b-2

contours Man image

long contours Man image



contours Rhinos image



long contours Rhinos image

Figure.15 contours and long contours for Man and Rhinos images

Edge extension is required because it extends the binary edge to the image boundary, improving the edge detection. After SE detection, some binary edge points are discontinuous, edge extension can complement the missing points between edge points, improving the edge detection.

b-3

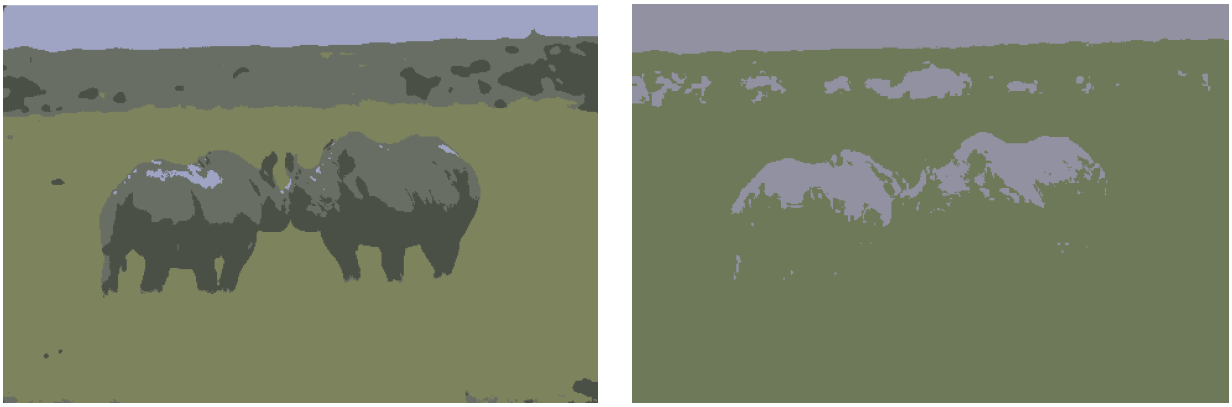
The purpose of selecting color samples along long contours is to perform **1D sampling** instead of 2D sampling. It results in a much smaller color samples set (image representation), simplifying the computation. Since using **contour-guided color samples** can perform mean shift in the spectral (range) domain, there is no need to do mean-shift in the spatial domain. Also, the pixels near the detected long contours are very useful for segmentation.

b-4

Figure.16 Man and Rhino images using their corresponding color palettes after quantized (radius=5)

Spectral radius parameter is the bandwidth of mean-shift clustering for pixels in range domain. It can determine the segmentations color in the image. As shown in Fig. when radius increases, the variety of colors decreases. More and more similar regions are assigned with the same color, since they are belonged to the same clusters in the range domain. For example, in Man image, as the man's red lip region is assigned with the same color as his face when radius become larger. It leads to the loss of information and objects in the image.

radius=10 radius=15 radius=20

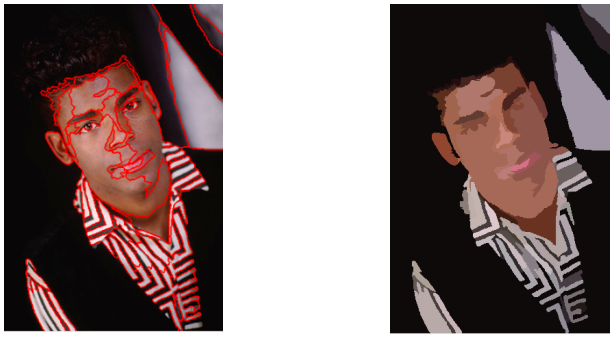


radius=10 radius=15

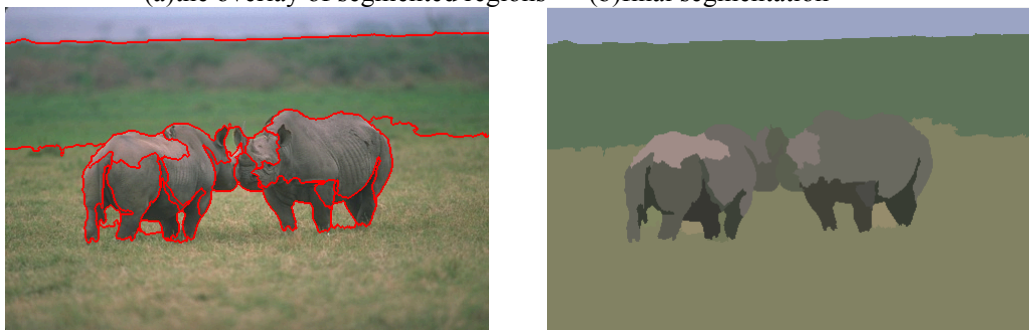
Figure.17 Man and Rhino images using different radius

Since the **K value is unknown**, it is hard to use **K-means** color clustering algorithm instead of MS algorithm.

b-5



(a)the overlay of segmented regions (b)final segmentation

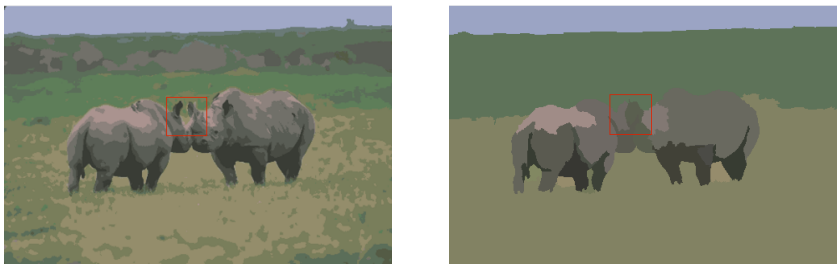


(a)the overlay of segmented regions (b)final segmentation

Figure.18 improved the final segmentation result by post-processing

There are three post-processing steps, **leakage avoidance by contours**, **fake boundary removal**, **small region mergence.**

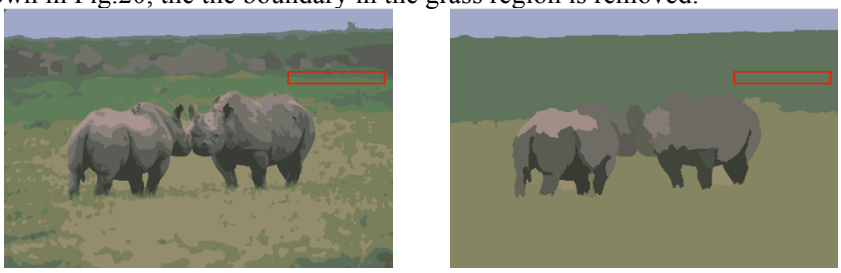
leakage avoidance by contours: since the colors on both side of contour can be similar, the edge of the segmentation will be destroyed. So we need to check the regions along each side to make sure the complete of contours. As shown in Fig.19, the leakage colors from two rhinos are avoided by checking the contour.



before after

Figure.19 leakage avoidance by contours effect

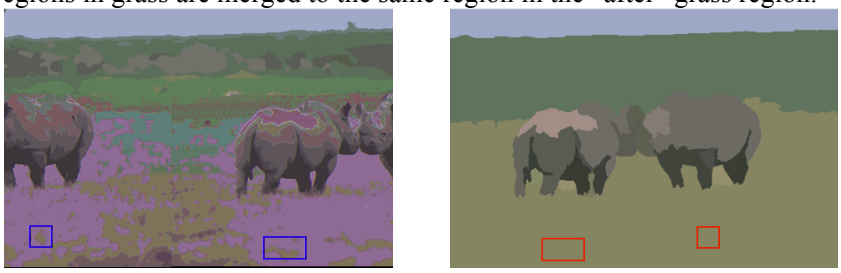
fake boundary removal: in some big region, the color transition is vey smooth (e.g. the sky, the sea, etc.). So we can check the common boundary of adjacent regions and eliminate this boundary (called fake boundary). As shown in Fig.20, the the boundary in the grass region is removed.



before after

Figure.20 fake boundary removal effect

small region mergence: some patches contains no useful information compared with their neighbors and we can merge them to the surrounding regions (or background), creating more homogenous regions. As shown in Fig.21, the small regions in grass are merged to the same region in the "after" grass region.



before after

Figure.21 fake boundary removal effect

c-1

Purpose: The measurement of image segmentation is difficult to measure. There is no common algorithm for the image segmentation. The statistical measurements could be used to measure the quality of the image segmentation, including COV, PRI, VOI, GCE and BDE.

COV (Segmentation covering) measures the overlap between two regions in computed segmentation and groundtruth. It is used for pixel-wise comparison of two images. $COV = \frac{1}{N}\sum_{R\in S}|R| \cdot \max_{R'\in G}\frac{|R'\cap R|}{|R'\cup R|}$, where S is the computed segmentation and G is the groundtruth. R and R' are the regions in the image. N is the total number of pixels in the image [4].

PRI (Probabilistic Rand Index) measures the similarity of two data clusters of two images. It counts the fraction of pairs whose labeling are consistent between the segmentation and the groundtruth images. The value $PRI = \frac{A}{A+D}$, where A stands for the agreements between computed segmentation and the groundtruth and B stands for the disagreements between these two images. So the lager the PRI is, the better the segmentation is.

VOI (Variation of Information) measures the distance between computed segmentation and groundtruth as average conditional entropy of the computed segmentation given the groundtruth. So it measures the difference

(randomness) in the computed segmentation based on the groundtruth. $VOI(A \parallel B) = H(A) + H(B) - 2I(A, B)$, where A is computed segmentation, B is the groundtruth. H stands for entropy and I is the mutual information between segmentation and groundtruth, so the smaller the VOI is, the better the segmentation is. And we will get higher mutual information.

GCE (Global Consistency Error) measures how much the computed segmentation is refined by the groundtruth. If the sets of pixels in a region in computed segmentation is proper subset of the same region in groundtruth, then it's well refined and the error is zero. If there exists overlapping in these two regions (not subset relation), it's badly refined (inconsistent). So the smaller the GCE is, the better the segmentation.

BDE (Boundary Displacement Error) measures the average displacement error of the boundaries pixels in the computed segmentation and the closest boundaries pixels in the groundtruth [5]. So the smaller the BDE is, the better the segmentation is.

I compare the two algorithms discussed above, mean-shift segmentation and CCP. Then calculate the quantitative performances below:

Man	COV↑	PRI↑	VOI↓	GCE↓	BDE↓
MS in part a	0.0864	0.9034	7.4492	0.8012	13.92
CCP in part b	0.3417	0.9117	3.8703	0.5225	9.36

(a)man image using different algorithm (MS and CCP)

Rhinos	COV↑	PRI↑	VOI↓	GCE↓	BDE↓
MS in part a	0.2248	0.8076	7.9921	0.6387	16.65
CCP in part b	0.4871	0.8855	2.1756	0.1355	10.69

(b)rhinos image using different algorithm (MS and CCP)

Figure.22 quantitative performances for MS and CCP algorithm

From the metrics above, we found that CCP improves the segmentation result compared with MS method. The computed segmentation has more similarity with the groundtruth and less error.

References

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvm

[2] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 24, no. 5, pp. 603–619, 2002.

[3] Xiang Fu, Chien-Yi Wang, Chen Chen, Changhu Wang and C.-C. Jay Kuo, "Robust image segmentation using contour-guided color palettes," IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, December 13-16, 2015.

[4] Contour Detection and Hierarchical Image Segmentation P. Arbelaez, M. Maire, C. Fowlkes and J. Malik. IEEE TPAMI, Vol. 33, No. 5, pp. 898-916, May 2011.

[5] R. Unnikrishnan, C. Pantofaru, and M. Hebert, "Toward objective evaluation of image segmentation algorithms," IEEE Trans. Pattern Anal. Mach. Intell., vol. 29, no. 6, pp. 929–944, Jun. 2007.

[6] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. PAMI, 26(5):530–549, 2004

[7] X. Ren, C. Fowlkes, and J. Malik. Scale-invariant contour completion using cond. random fields. In ICCV, 2005