Image Segmentation based on Random Forest

Tian Ye: tye459@usc.edu; Shuo Wang: wang133@usc.edu

Abstract

Image segmentation refers to partition of an image into a set of regions that cover it, for example, most simple task is distinguishing object and background. Our team aims at higher goal, where single image may contain more than one object and complicated background. We implement two image segmentation methodologies based on [Texture analysis, SAAK Transform] via random forest on BSDS500 dataset. The overall algorithm includes pre-processing, feature extraction, training random forest, image reconstruction and post-processing. Our visual results of both methods are persuasive.

1. Problem Statement and goals

In short, we try to convert image from left to right as shown below:

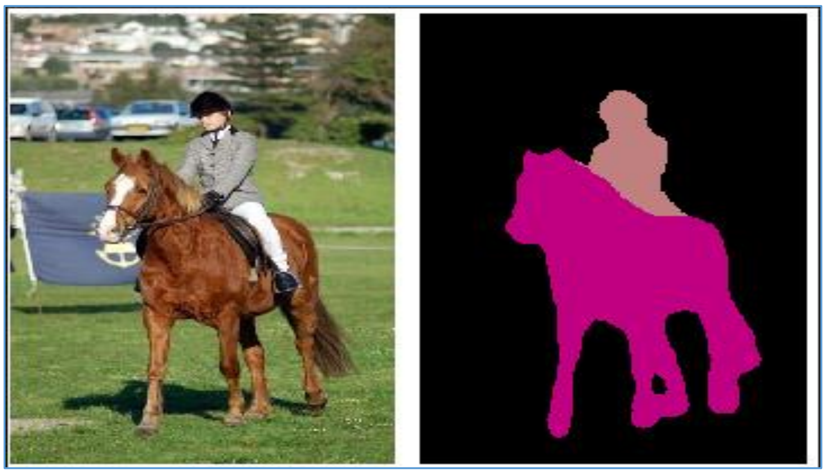


Figure 1 original and target image

Image segmentation is a typical classification problem, where input is pixels and output is respective labels. For training set, labels are given and are used to train ML model.

This is a tough task if we are not allowed to use any deep learning method.

First, there is no well-designed feature in dataset, and you can not just input single pixel's intensity for single pixel showing too less about the region so that you cannot extract any useful texture information. How to generate good feature is one of the most important factors in this project.

Second, mapping between input and output is clearly highly non-linear. Ideally, we need to apply multiple layers of convolution operation to dig the spectral information about the pixel yet texture analysis can just see shallow layer. That is one reason why we use SAAK transform, who has multiple layer structure, as plan B. In fact, we do not abandon texture analysis completely, experiment shows that with modified texture analysis, segmented image looks good as well.

Third, there are so many detail and adjustable parameters, including parameter about trees, reference window size, gray image or not, blur or not, reading with stride or not... Due to our limited performance laptops device and large data scale, tuning parameters can be extremely time-consuming, thus instead of just trying bunch of setting, we have to find a possible theoretical explanation of current insufficiency, and try corresponding improved methods.

2. Prior and Related Work

SAAK Transform from EE669

SAAK Transform is a newly raised image feature extractor, we are asked to apply it on MINIST classification problem in EE669. It is very similar to CNN for its multiple-layers framework except for it doesn't need any back propagation, one should collect output from each layer and concatenate them to form a new vector as feature vector. On MINIST dataset, it reaches 98.4% test accuracy. We have not any experience where applying SAAK on other task or dataset before, hence utilizing SAAK in our project is an exploring.

3. Project Formulation and Setup

3.1. pre-processing:

3.1.1 Reduce DC for every image

$$\mathrm{I(x,y) = I(x,y) -\ avg\{I(x,y)|I(x,y)in\ \ image\ \ scope\}}$$

3.1.2 Gaussian Blur

$$\mathrm{I = I\ \ conv\ \ \{Gaussian\ \ Kernel\}}$$

3.1.3 Build small patch for each pixel with stride n

Patch is centered by the pixel, suppose patch size is 3x3 and pixel coordinate is (x,y), then location $\{(x-1,y-1),(x-1,y),(x-1,y+1),(x,y-1),(x,y),(x,y+1),(x+1,y-1),(x+1,y),(x+1,y+1)\}$ form a patch. Stride means whether we load pixels with skipping, stride = 1 means no skipping, stride = 2 means skipping one position. For example, current read pixel has coordinate (x,y), stride = 1 means next pixel is $(x,y+1)$ while stride = 2 means $(x,y+2)$

3.2. Feature Extraction

3.2.1 Laws Filter Banks:

Baseline: Five by five Laws Filter bank are generated by five one-dimension filters:

$$\{L = [1,4,6,4,1], E = [-1,-2,0,2,1], S = [-1,0,2,0,-1], W = [-1,2,0,-2,1], R = [1,-4,6,-4,1]\}$$

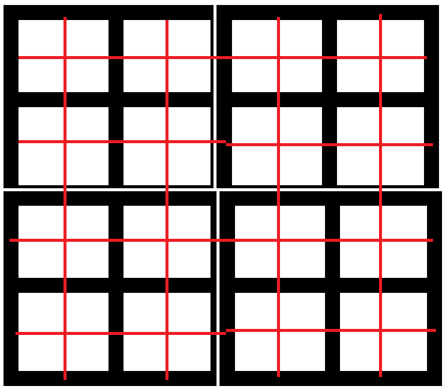
via $F[p] = f[i]^T f[j]$ where i and j in range 1~5.

Enhanced Feature: Beside texture information, we add color information at the tail of the vector. Notice, color information does not participate PCA and STD processing, it is additional feature. Apply those filters to image, convert to energy form and generate a 25-dimension vector for each pixel in each channel.

3.2.2 SAAK Transform:

SAAK transform can be divided into two steps: SAAK transform and inverse SAAK transform. In this question, only SAAK transform will be considered. After applying SAAK transform, the spatial size of input will be decreased. Here is the method of one step SAAK transform.

- Divide the original input image into a series of $2 \times 2 \times \mathrm{C}$ blocks if original input is $\mathrm{M} \times \mathrm{M} \times \mathrm{C}$, the number of which is $M/2 \times M/2$. The division method can be shown as follow:



- For each little block, reshape it into a 1-D vector, meaning that those blocks can form a 2-D matrix.
- For the 2-D matrix, do the PCA to decrease each vector dimension from C dimension to F dimension.
- Reshape the output 2-D matrix above to make the shape of matrix to be $M/2 \times M/2 \times F$. Then, for the first F – 1 channel of pixel, it will time -1 and be connected to the tail.
- For each entry of the matrix, if it is larger than 0, the value will be preserved; else, it will be set to be zero, which is the final output of SAAK transform.

The reason to apply SAAK is that the SAAK transform and inverse transform is a link between the spatial message and spectral message. When applying SAAK transform, local adjacent message will be concentrated and message will be saved as the pixel point. As a result, it there is a leaf in an image, it cannot be represented as the leaf if only checking one point since the leaf has several pixels in the image. However, it will be classified as image according to one-point message if applying SAAK to concentrate the energy in several points. For the inverse SAAK transform, the spatial relation will be built from the spectral message. As a result, the spectral result will be visualized because of such method.

3.3. Machine Learning Algorithm:

3.3.1 K Means cluster:

we do not use k means just for performance comparison but for inevitable reason. This is due to in BSDS500, ground truths of images are not generated via texture but more like object based. For example, there are many texture types on a tiger but tiger are labeled as one object. If we use random forest in this case, no way you can just use texture as criteria. Thus there are two methods, either we try to enhance texture features or quit supervised learning. At the very beginning of the project, we do not think about anything about unsupervised K-means algorithm.

K Means algorithm can be briefly described with following pseudo code:

```
K-Means
{
    Randomly generate K initialization centers;
    While (do not coverage)
    {
        Assign all points to respective nearest center;
        Use mean of points in a cluster as new center;
    }
}
```

Notice：K in our program is decided by label analysis module, it is not a adjustable parameter.

3.3.2 Random forest:

Random forest is based on decision tree, each node split current cluster to two clusters via setting a threshold on certain feature. Random forest consists many decision trees, they vote to get final label. The idea is although there are so many weak decision trees, there should be at least some good individual trees. Weak trees product random wrong outputs which will counteract themselves and good trees' result will be peaked as selected as output. Compare to other machine learning model such as linear regression, random forest does not have loss, lower opportunity to over-fitting with more trees and limited depth, and most important point is it can be applied to real task easily, more like a general model.

There are mainly 5 key parameters in random forest model: number of trees, maximum depth, minimum sample split, minimum sample leaf and number of features. Usually we tend to more trees, lower depth, and appropriate sample split, sample leaf and number of feature according to specified problem and feature.

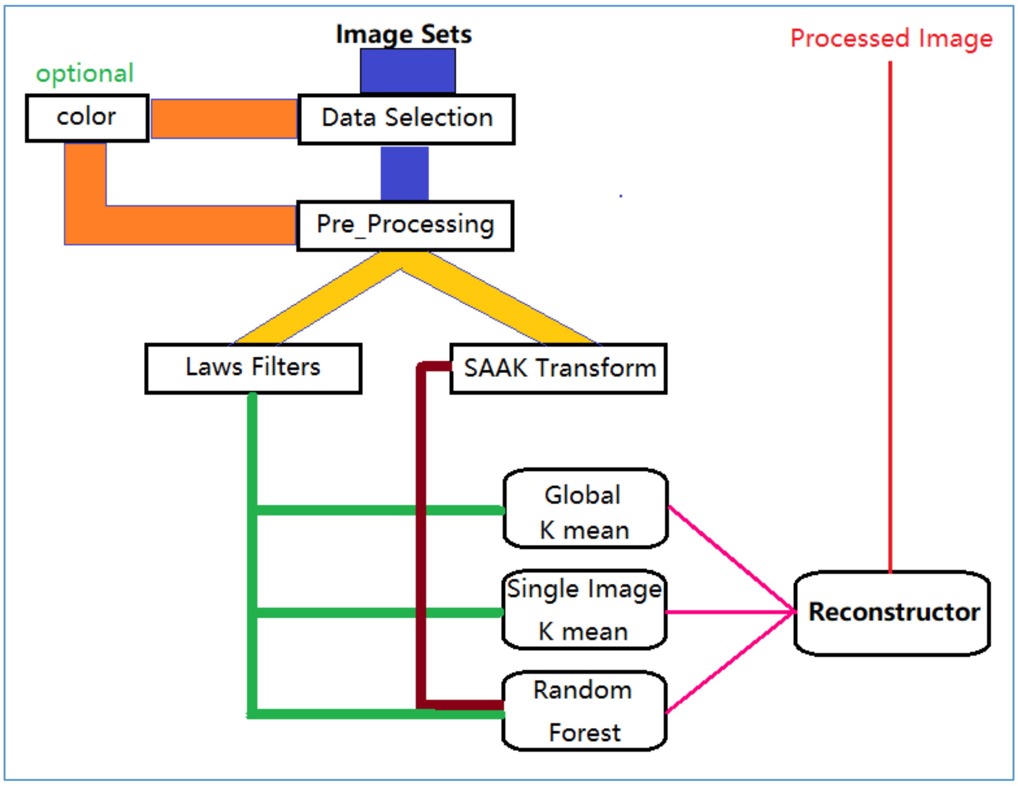
3.4. Reconstruction

Reconstruction is to acquire the segmentation result for a raw input images, the process can be shown ass follow:

- For an image, apply Gaussian filter to blur the image.
- For each pixel, consider surrounding 7×7 pixels and resize it to be 8×8 (SAAK) /surrounding 5 x 5 pixels (Laws Filters).
- Then, transformation will be applied to make the input features fit the input requirements of Random forest.
- Input the transformation result into random forest classifier and get the prediction result.
- Reshape the previous result into image shape, which is the final output of segmentation.

4. Methodology

The basic flowchart of process can be shown as follow:



The basic method for image segmentation based on random forest can be shown as follow:

- **Load the BSDS500 datasets**
  The dataset includes 200 training images and 200 testing images whose size is $321 \times 481 \times 3$ or 481 x 321 x 3(RGB color image) and relevant ground truth matrix whose size is 321×481 or 481 x 321.

- **Acquire raw input and output pairs**
  Laws Filters based: Due to size of Laws Filters kernels are 5 x 5, hence we use surrounding 5 x 5 window to present a pixel.
  SAAK Transform based: For each pixel in the image, surrounding 7×7 will be obtained to be considered as the raw input for random forest and relevant ground truth pixel is the output label for random forest. This process will be applied in both training and testing datasets

- **Choose of samples**
  Since the frequency of occurrence for different labels is severely unbalanced, we have to choose the sample seriously.
  First, filter those labels whose frequency is less than a threshold (Laws: 10000 SAAK: 20000).
  Then, for each label, randomly choose specific number (20000) of samples from each label.
  Having applied such procedure, each class can be considered evenly, which is good for classification for different texture.

- **Data transformation**
  It will be divided into 2 cases, where both training dataset and testing dataset will be utilized:

Law's filters based: we filter an image to get 25 response map. Then for each pixel, we check surrounding 5 x 5 window and compute their average energy. If input is gray scale image, then a pixel is presented by 25-values vector; if input is RGB image, 75-values vector will be applied.
SAAK transformation based: input image will be firstly resized into 8×8 and do SAAK transform to get the SAAK information of the image. Then, PCA will be applied to coordinate the dimension of input to be the one desired.

- **Training of Random forest**
  In this section, the random forest will be applied to do the classification. The hypothesis set of random forest is random forest with different structured tree depth, different minimum number of samples in a leaf node, different minimum number of samples that can be divided in the node and different maximum of features that will be considered during a division.

- **Training of K-Means**
  There are two possible methods to use K-Means. One is online-training and aims at single image, thus actually do not need no regular training procedure. Another one need all train data to generate model (K center points) and test image's pixel will be assigned to corresponding nearest center's respective label, hence it is offline-line training. We implement both methods, detail will be discussed later.

The evaluation can be divided into 2 steps: MSE and visual evaluation.
For the testing window patches, the MSE will be calculated. The reason to calculate the MSE is that the close label will represent the similar color and texture, meaning that it will be a bit difficult to classify those features significantly. Meaning that MSE will be more reliable to test the training and testing result, which can be shown as follow:

$$\mathrm{MSE} = \frac{1}{N}\sum_{i}(output_i - label_i)^2$$

The closer MSE is to 0, the better the segmentation effect.
For the visual evaluation, justify whether different color and texture will be divided properly according to the input.

5. Implementation

Based on the methodology above, there will be some significant part to be specified.

5.1. Feature space

In the segmentation, the BSDS500 dataset will be the raw dataset. The input image is 200 $321 \times 481 \times 3$(RGB) training images, 200 $321 \times 481 \times 3$(RGB) testing images and the output matrix is relevant 321×481 matrix relevant the texture of the pixel.

As a matter of fact, it is not the final feature for the random forest, which will be acquired by feature preprocessing and feature extraction. However, for the label result in the output matrix, there is a pixel in the original image to point. The relevance can be shown as follow:

Feature name	Data type	Range
R	Uint8	[0, 255]
G	Uint8	[0, 255]
B	Uint8	[0, 255]

The feature extraction method which will be applied in random forest will be shown in the next section.

5.2. Pre-processing and feature extraction

The progress to acquire the feature which can be applied for random forest can be shown as follow:

- **Image preprocessing**
  For the raw image, minus 128(SAAK) or mean of image (Laws Filters Bank) and pass the Gaussian filter and Median filter to blur the high frequency element which is not required for segmentation task.

- **Input-output pair acquirement and selection**
  Even though each pixel has a relevant texture score, surrounding 7×7 pixels (SAAK) or 5 x 5 pixels (Laws Filters bank) will be considered to map the relationship between the input and output. In addition, surrounding 7×7/5 x 5 points in output matrix will be acquired relevant to 7×7/5 x 5 input image patches, which will form input-output pairs. Then, filter those pairs whose output has more than 1 kinds of labels. The final output is the central point value of 7×7/5 x 5 texture points.

- **Sample choice**
  Since the number of each labels is different, whose distribution is severely unbalanced, we have to choose samples according to the section, which will acquire 25 class with 20000 samples in each sample.

- **Data transformation**
  This process will be considered according to different methods:
  - For Laws filters, we have tried: RGB texture (doesn't work), Gray texture (doesn't work), Gray texture with addition color information (work). Each channel will have 25 feature present low to high frequency on x and y axis. Color information will add 3 intensity of color channels, who are scaled to 0~1. For example, RGB texture will allocate 75 features to one pixel while Gray texture with color will assign 28 feature to one pixel.
  - For SAAK transform, resize the input patches into 8×8 and apply 3 cascaded SAAK transform, which will get output 4×4, 2×2 and 1×1. Then, concatenate them for each image, which will acquire both spectral and spatial messages. Then, PCA will be applied to fit the input size to enter the random forest.

5.3. Training process

As it is said above, for all training patches, there will be two steps to choose the patches that will be put into the classifier: firstly, the relevant ground truth patches with not only one kind of labels will be filtered since the texture in this patch has more than one kind. In addition, the sample number for each high frequency class will be randomly chosen to be same frequency. This process is the same as the training dataset and testing dataset. As a result, there will be 20000 samples for each label and 64 features for final input into random forest.

- **Random forest:**

For the training dataset, which has relevant input image patches and output labels, the Random Forest classifier will be applied to fit the dataset and do the reasonable prediction to the test dataset.

As is said above, the hypothesis set for the classifier model is the set of Random forest and its complexity is determined by different parameters, including different structured tree depth, different minimum number of samples in a leaf node, different minimum number of samples that can be divided in the node and different maximum of features that will be considered during a division. Those choice of parameter will be introduced in the next section, the Testing and Model selection.

- **K-Means:**

For K means, who uses all train data to build model, there actually not much hypothesis to choose from. First, K is defined by how many valid labels (filter those have too less samples) are there, not via randomly trying a number. Second, although we can set different halt condition and maximum iteration steps, our experience tells us usually K means converges pretty first and default setting 300 iterations is good enough for most cases. Hence we skip K means in next section.

5.4. Testing and Model Selection

For the Random Forest, there will be five parameters to be considered:

- structured tree depth
- minimum number of samples in a leaf node
- minimum number of samples that can be divided in a node
- maximum of features

As a matter of fact, each parameter except second and third one, which will be considered at the same time, will be considered independently.

For the depth of tree, the relationship between depth and training and testing MSE can be shown as follow:

Depth	5	9	13	33	57
Training MSE	102.797	82.415	80.696	80.537	80.621
Testing MSE	131.077	104.388	105.76	106.211	105.339

From the result above, it can be found that the depth of tree can decrease the training MSE while the testing MSE will not decreased when the depth of CART is 13.

For minimum number of samples in a leaf node and minimum number of samples that can be divided in a node, I will consider them at the same time. As a matter of fact, if a node can be divided, it can be divided into two parts with equal number of samples, and those child nodes can form the leaf nodes. As a result, the relationship between min_samples_split and min_samples_leaf can be shown as follow:

$$\text{minsplit} \geq 2 \times \text{minleaf}$$

In this case, I set $\text{minsplit} = 3 \times \text{minleaf}$. In addition, because the complexity of random forest is inversely proportional to the number of samples to split or to be leaf. I set an initial number of samples to split or to be leaf, which is 10000 and 3000. For less number cases, it will be represented by $\text{minsplit}/N$ and $\text{minleaf}/N$, meaning that N will be the variable for complexity. The relationship between N and training and testing MSE can be shown as follow:

N	1	3	9	11	13
Training MSE	80.383	66.86	65.288	62.118	60.380
Testing MSE	109.102	102.87	122.324	121.54	123.197

From the result above, it can be found that when N is larger, the training MSE will decrease significantly while the testing MSE will decrease first and then increase. Therefore, N should be chosen to be 3, meaning that the minsplit is around 3000 and minleaf is around 1000.

For the maximum feature, I tried 32, 48 and 64 as the parameter, the result can be shown as follow:

Max Features	32	48	64
Training MSE	66.237	63.445	66.798
Testing MSE	111.145	109.438	102.55

From the result above, it can be found that applying all 64 features will acquire the minimum training and testing MSE.

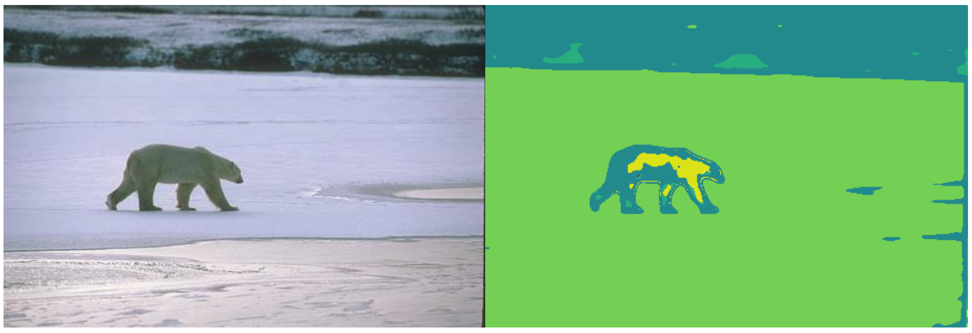
As a result, I set depth to be 11, minsplit is around 3000, minleaf is around 1000 and max features is 64. In this question, 200 trees will be built to do the classification, the training and testing result is 60.862 and 81.016.

6. Final Result

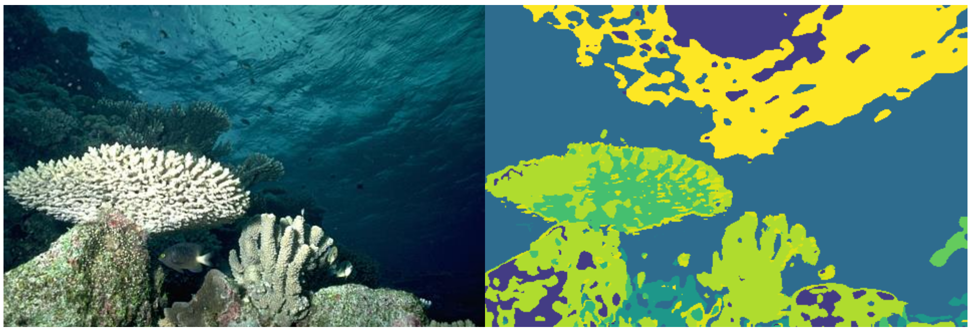
(i) SAAK Transform with Random Forest:

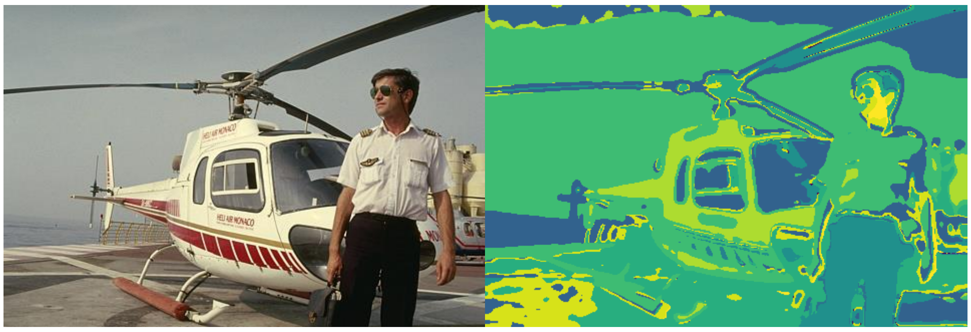
I applied the random forest classifier to reconstruct the test images, result can be shown as follow:

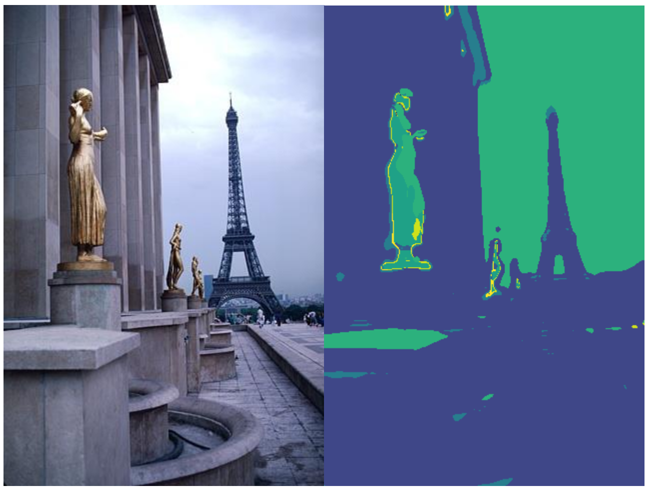
Test fig 100007

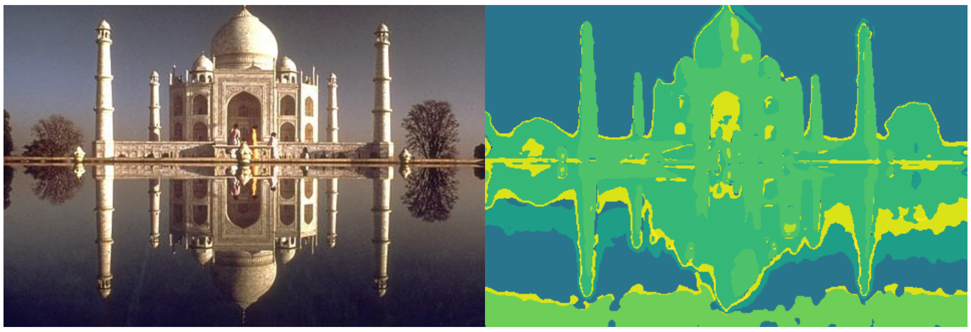


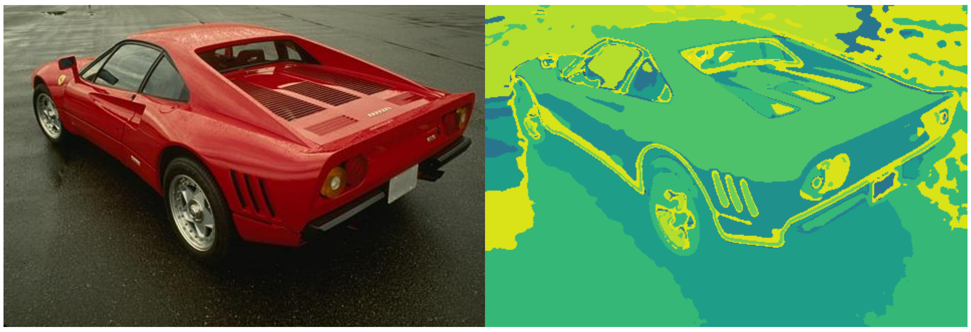
Test fig 101027

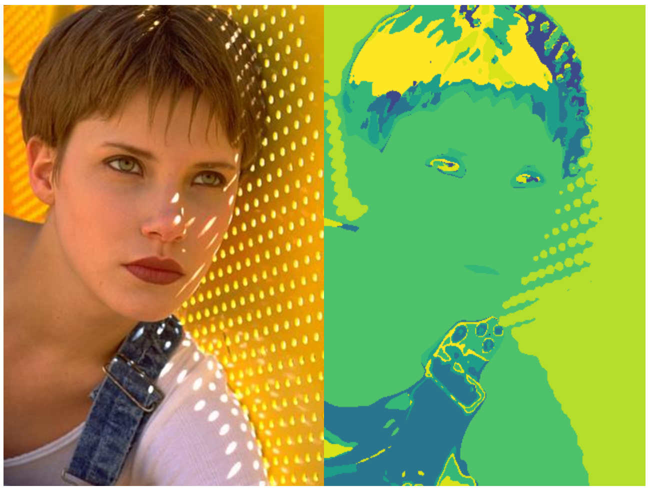

Test fig 146074

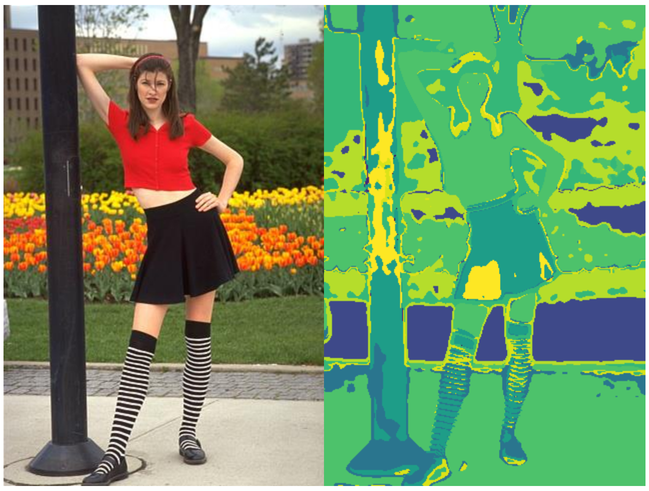

Test fig 223004


Test fig 288024


Test fig 29030

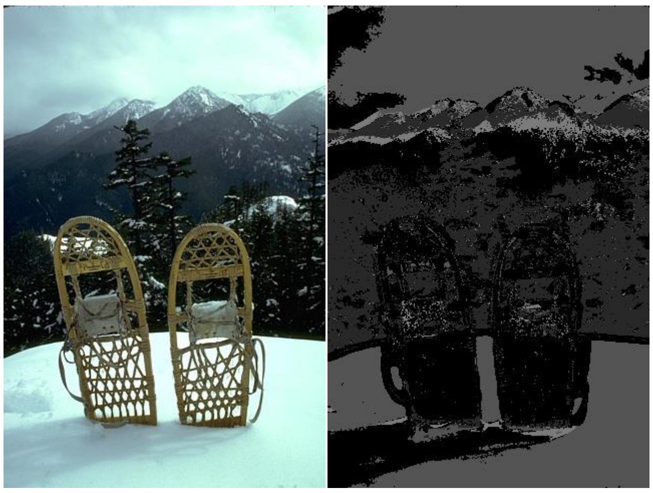

Test fig 388006

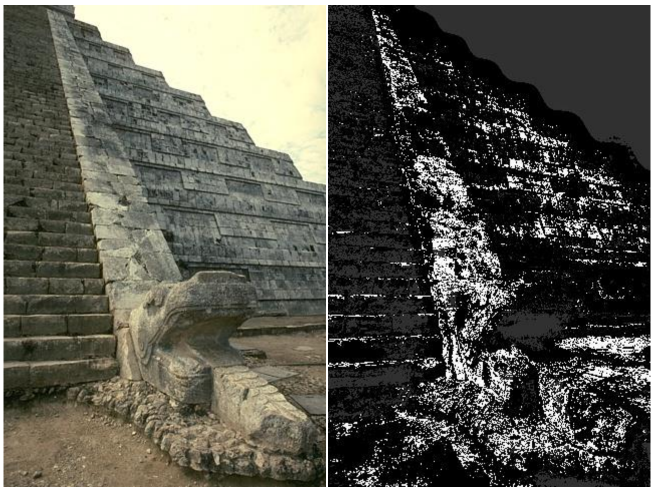

Test fig 388067

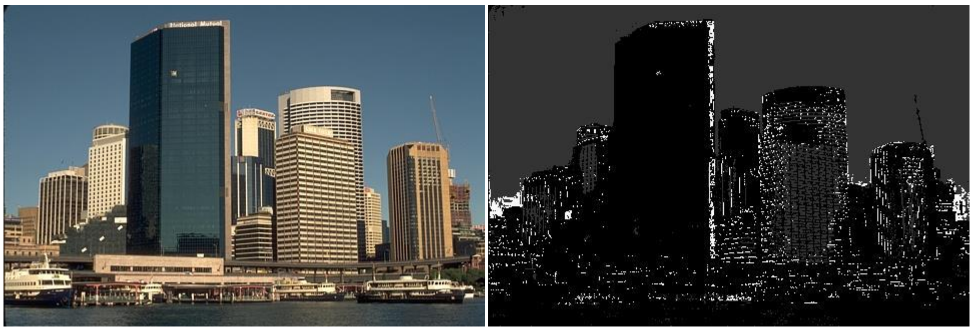


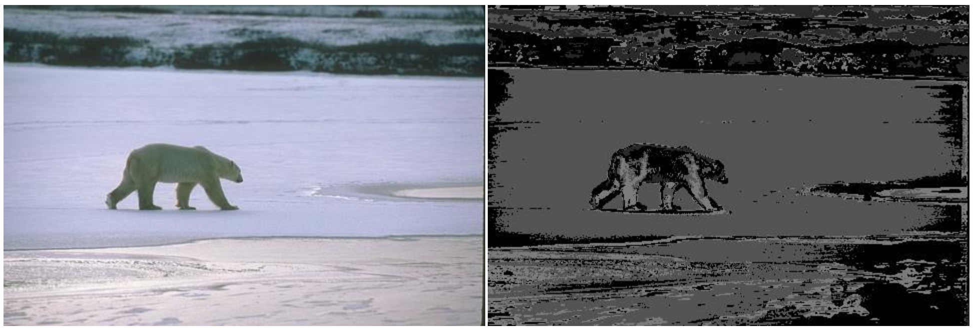
(ii) Laws_filters_Banks with color and Random Forest:

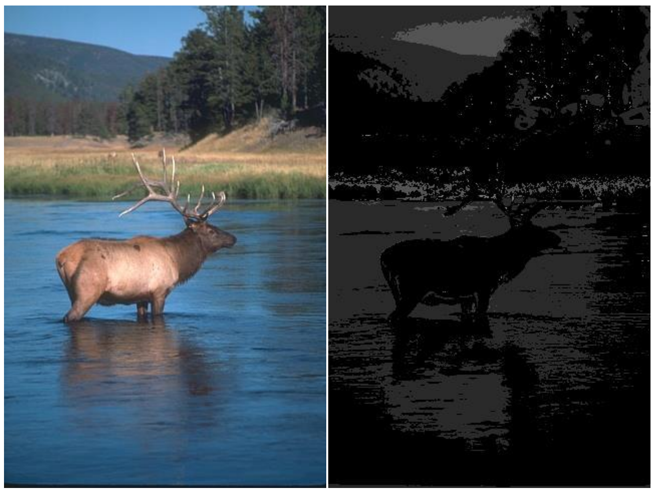
Test fig 2018

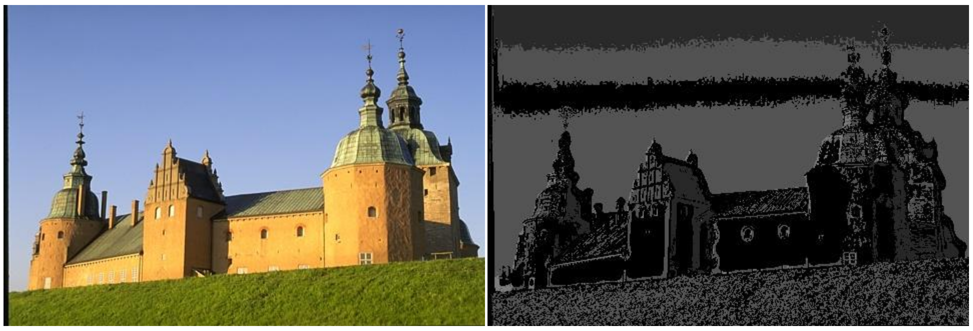

Test fig 33044

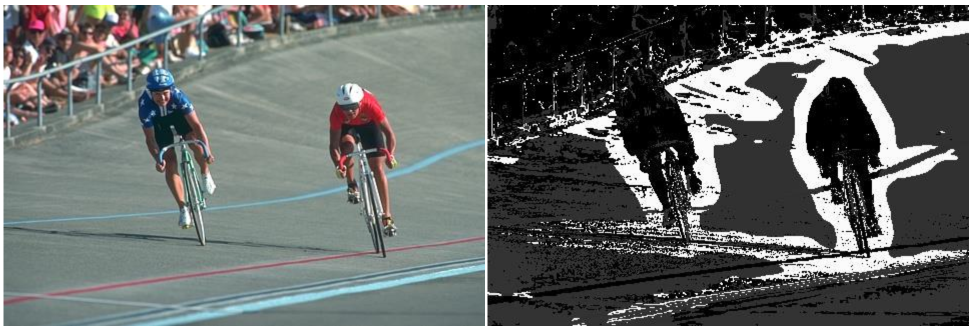

Test fig 69007

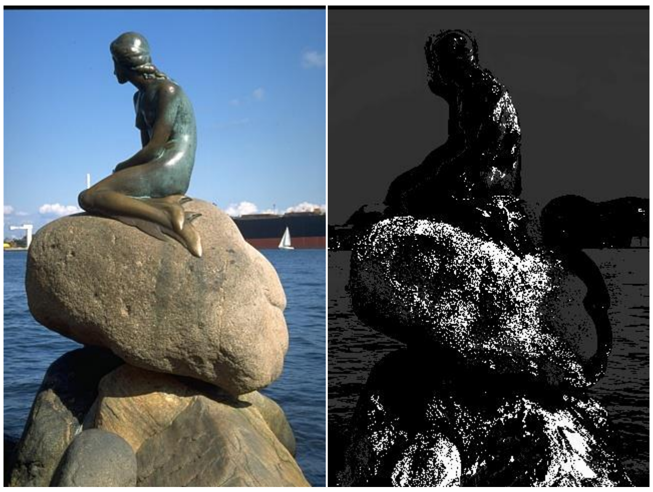

Test fig 100007


Test fig 104055


Test fig 201080

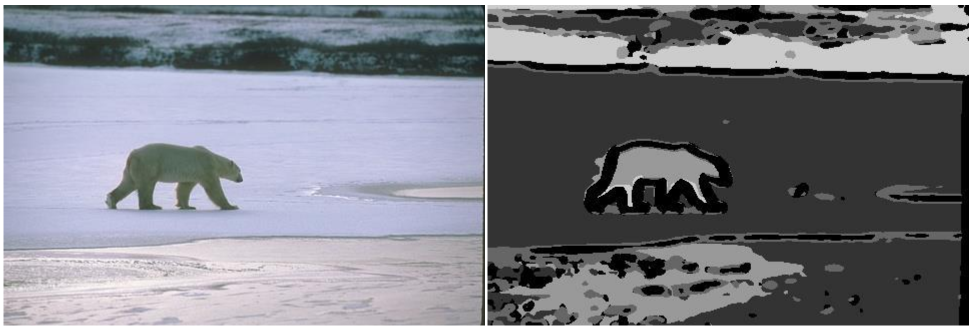

Test fig 226022


Test fig 372019

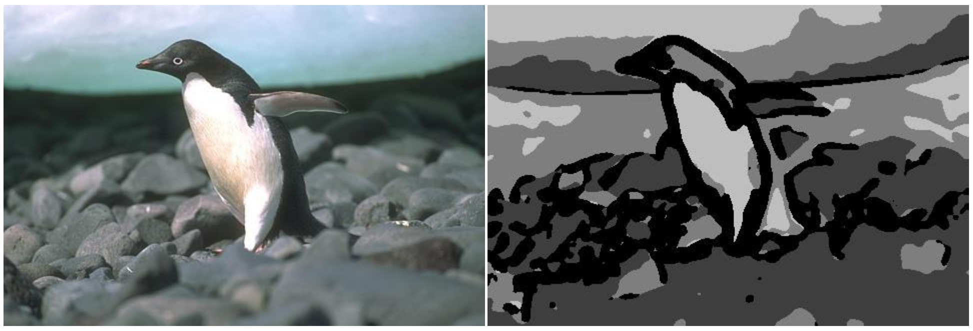


(iii) Laws_filters_Banks with color and global k_means:

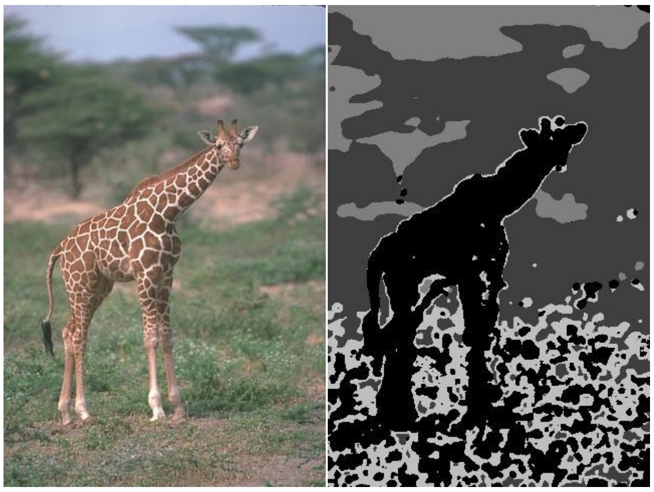
Test fig 100007



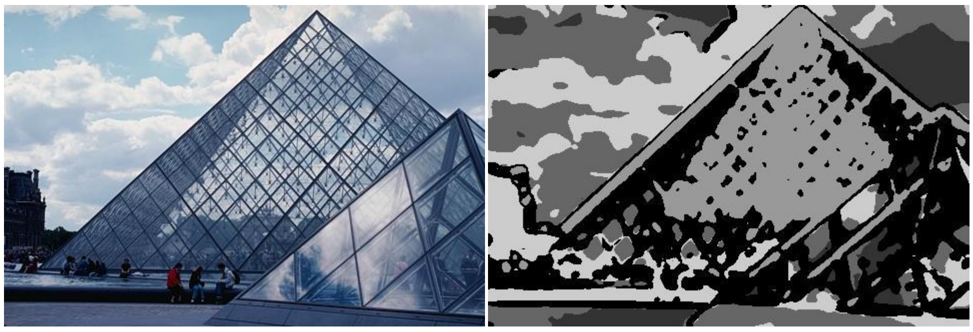
Test fig 106005



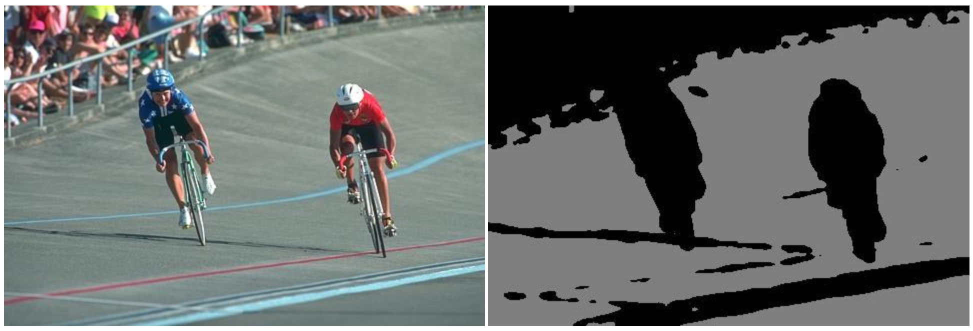
Test fig 130014



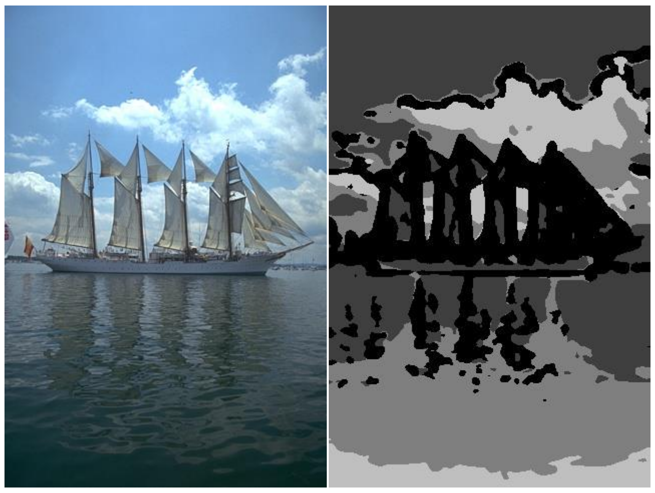
Test fig 223060



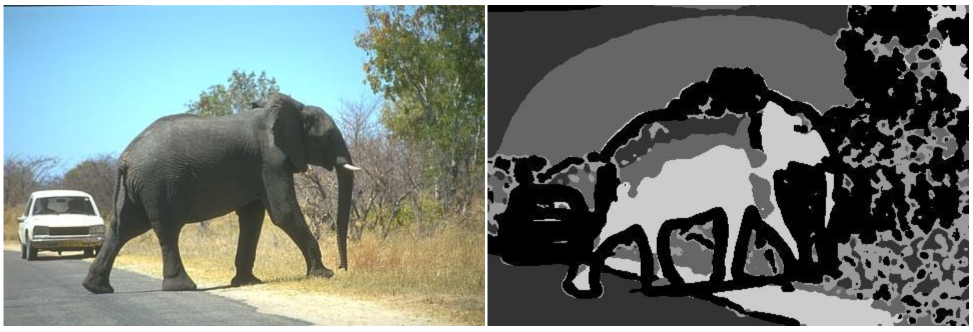
Test fig 226022



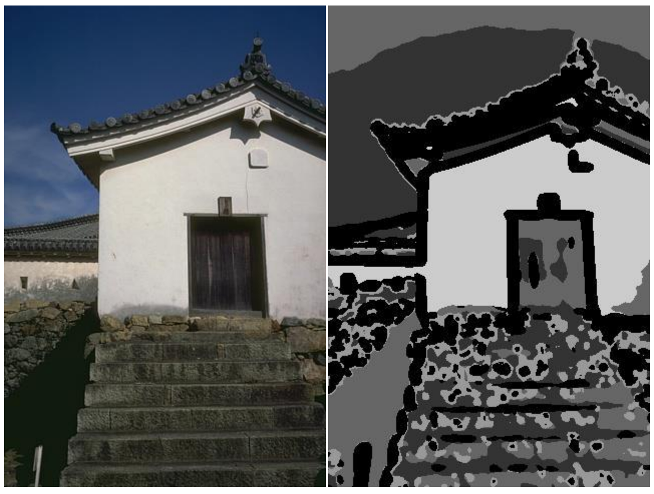
Test fig 257098



Test fig 296028

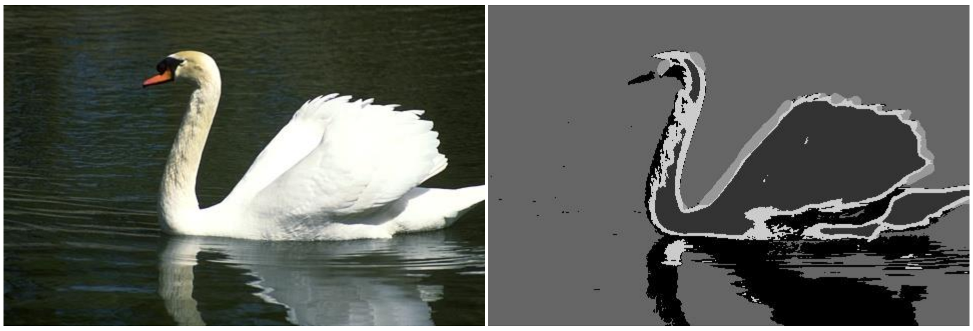


Test fig 334025

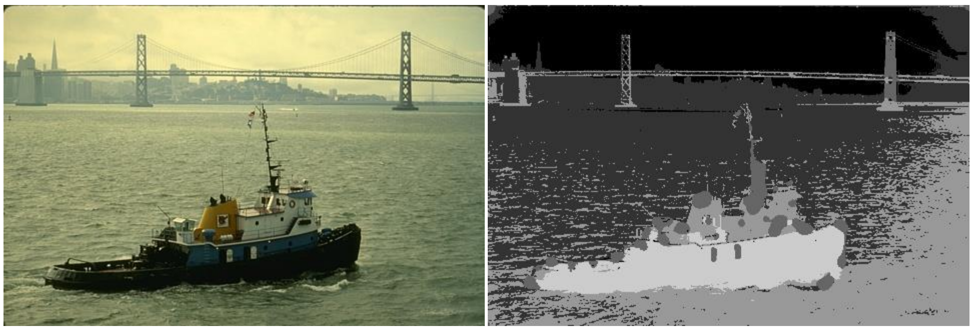


(iv) Laws_filters_Banks with color and single image k-means:

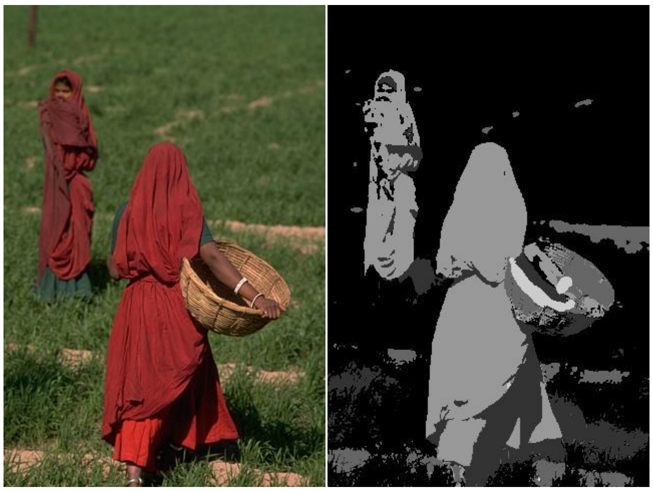
Test fig 8068



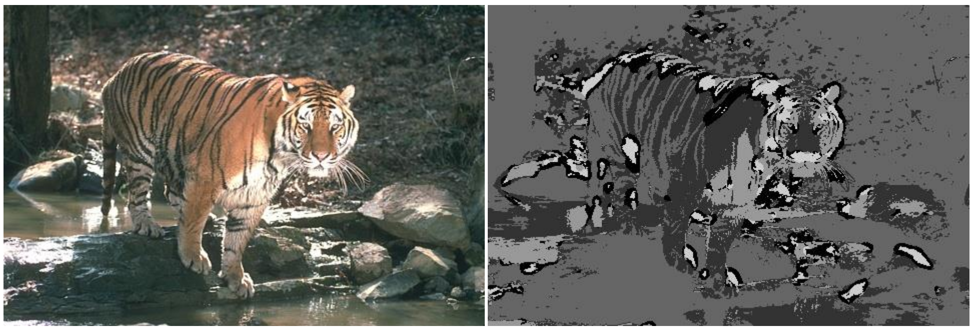
Test fig 48025



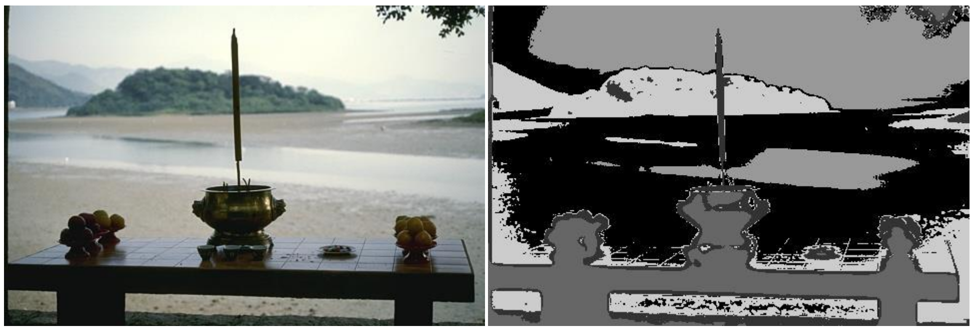
Test fig 80090

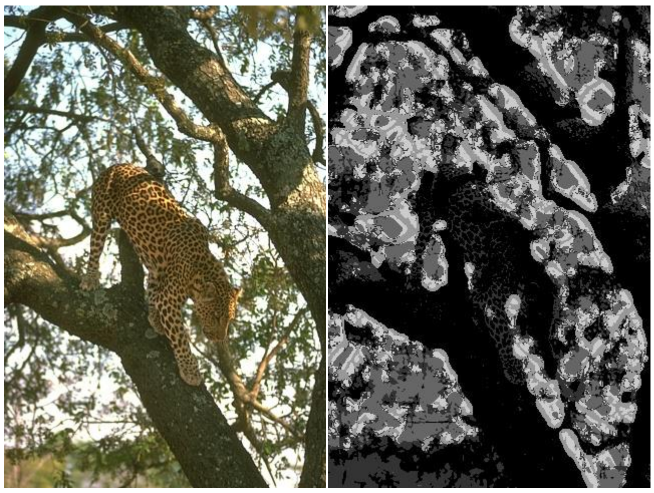


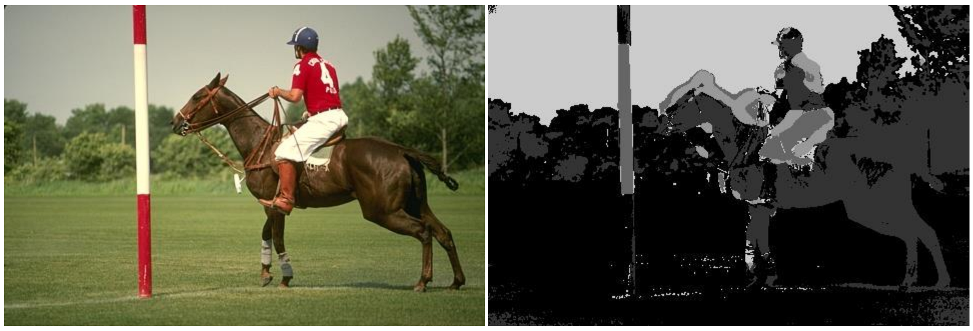
Test fig 108036

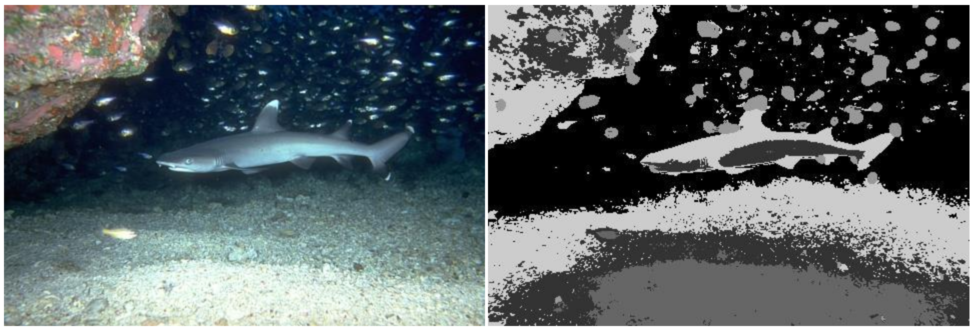


Test fig 12003


Test fig 134067


Test fig 206062


Test fig 306052



7. Interpretation

(1) Laws_Filters part:

Why concatenate color information?

At the beginning of the project, we apply laws filters to get texture information and use texture and labels from ground truth to train random forest. However, outcome is pretty pathetic. Processed images are filled with fragile blocks and you cannot distinguish anything no matter how we tune trees.

We believe the problem lays on object based ground truth. Laws filters can just extract feature information, yet ground truth is segmented by object. For example, tiger may have several texture (body, tail, head, and so on), thus ideally ground truth should label those textures with different index. However, due to human label things according to object, those textures just have one label. Hence by doing above operation, we send many wrong and confusing data to random forest. As result, model may allocate the same smooth region with different labels, that's the reason why there are so many fragile blocks.

To improve, we need more persuading feature, enlarging kernel size (so we can extract more global information) may be reasonable yet the size of Laws Filters is fixed. Hence we add color information behind texture feature, although it is not optimized solution but we do find better visual result. Processed image still has a bunch of noise but those noise is small enough to be eliminated by post processing (this also depends on how complex original image is).

Why use global/single K-mean?

As mentioned above, objected based labels are conflicted with texture based feature, hence why not just abandon labels? Result shows that unsupervised K-mean algorithm obtains better performance than random forest. In addition, K-mean is not sensitive to data quantity distribution. Theoretically speaking, similar vectors will gather together in the end no matter how many samples are there. Global K-mean still need training dataset to train model, while single image K-mean don't even need large -scale training. Due to scope of single image is small, thus number of labels in each image is small, low complexity data means easier to get good result, in our results, single image K-mean does have most stable and best visual result.

(2) SAAK part:

If we still want to use random forest, more powerful feature extractor is needed. In EE669, we learn that SAAK transform can product "object information" on MNIST dataset pretty well, thus assuming SAAK transform can product matched feature with ground truth on BSDS500 is reasonable.

The reason why SAAK is useful to process patch is that SAAK transform will decrease the spatial relationship and create more spectral relationship. In the spectral relationship, the frequency value will be preserved, meaning that the texture information will be saved. Based on the mapping between the texture information by SAAK, it can help the classifier to do classification according to the texture and color of the image.

(3) Random forest

Even though the definition of random forest is easy for us to understand, the parameter to coordinate is a serious question since there are more than 4 parameters that can affect the training and testing accuracy significantly. When I applied different value for a specific parameter, I got a MSE curve which is really relevant to the relationship between E_{in} and E_{out} in EE660. In addition, I have understood the parameter's meaning during applying those random forest by combining theory and practice. For a single decision tree, its power is limited. If we have 1000 of them, assume 900 of them are weak and more likely to product wrong outcomes, and only 100 of them can product correct answers. One assumption is that those 900 wrong predictions will be pretty even so that 100 correct outcomes will be peak. If there is not enough trees in the forest, there may not be such peak, thus make less overall performance weaker. However, too many trees may cause serious performance problem, taking us forever to train, thus usually we need efficiency as a trade off with performance. To feature numbers, we believe it depends on how important is each dimension. In our experiment, final feature vector has been through PCA and some other modification, hence every dimension is essential, we cannot just use sqrt(N) as feature length. Tree depth effect over-fitting problem a lot, this parameter manly depends on how complicated data is, what we can do is trying multiple value and select the best one.

(4) Unexpected Issues

- **Poor efficiency of python if trying to code everything**
  As a matter of fact, we were not familiar with python library, especially the libraries based on Numpy. If you keep normal C/C++ language program style (a lot of loop, implement everything from baseline...), coding efficiency will be horrible. At first, we even tried to extract image patches based on the C/C++ code (we do implement feature extracting part with C/C++). However, it is pretty annoying to complete single task on different platforms, where we have to import/export data a lot, thus we eventually stick to python platform. Later when we dealt with the experiment, we consult the functions in libraries based on Numpy, which is a huge work for us. Yet now we can say that we feel comforting write python script.
- **Underestimate the size of data**
  This is a common problem in machine learning. At first, we do not think that the size of data is a serious issue for us. However, when extracting the image patches, it can be found that the memory of computer cannot save so many data, meaning that it will be impossible for us to calculate all the data required. As a result, having acquired the raw data, I increase the stride step in order to decrease the training samples, which will satisfy the memory requirement.
- **Severely unbalanced label type distribution**
  Before this project, dataset we use in other machine learning problem are "well designed", samples are uniform distribution. In BSDS500, we even find some sample type who just has Less than 10 instances. If there are bunch of those small sample types, it may have bad effect on random forest. For example, it may need higher depth so that those small groups can be separated from other cluster, which may cause over-fitting problem. After we filter small groups and select data uniformly, apparently less noise in processed image.
- **Unmatched feature and label**
  Our original plan uses texture but labels are based on object. We solve this by three methods: modifying texture feature; using unsupervised machine learning algorithm and apply SAAK Transform to get matched feature.

Overall conclusion is in section #9

8. Contributions of each team number

Although we complete project as team, our cooperation is limited to discussing idea and analyzing current result. Both of us implement Laws Filters based and SAAK based (with random forest) segmentation (check our code). Basically after we get an idea, we write our own code, then share and discuss. In the middle of the project, Tian Ye agrees to focus on Laws Filters while Shuo Wang focus on SAAK Transform. The report is written by both of us.

9. Summary and Conclusion

Overall, we have tried both supervised learning and unsupervised learning method to do the image segmentation based on the texture patches. From our perspective, these methods can be useful to classify different texture effectively and make the segmentation according to the texture and object accurately. However, it will be hard for the case where foreground and background texture is similar. In addition, the label which is provided by ground truth is not really reliable, which affects our final output to be not pure.

- **What is the key in conventional machine learning?**
  In almost the whole semester we were talking about models related stuffs like object function, optimizing approach, sparsity... But in real project, we think the key lays on features. Good feature or matched feature is the foundation of machine learning. In our experiment, early trial use unmatched feature and result is a mess; later we used better but not good enough feature, quality improved but still have a lot of noise; then with optimized feature from SAAK Transform or unsupervised machine learning algorithm, outcome becomes high quality.

- **Model complexity should match with data complexity**
  For the random forest, there will be a series of parameters which can affect the model complexity, especially the depth of CART, minimum number of samples to split in a node and minimum number of samples in a leaf node. As is said in previous section, the deeper the CART is, the more complex model will be; the less the minimum number of samples to split is, the more complex the model will be, which is similar with the minimum number of samples in a leaf node. According to experimental result, it can be found that if the model is too complex, the model will be over-fit, meaning that the test accuracy will decrease while the model is to simple, it cannot represent the relationship between input and output correctly.
  For K-means algorithm, the number of cluster is the key for the model complexity. If it is really large, some single label will be regarded as multiple labels; if it is really small, many class cannot be clarified by the classifier.

- **Sometimes, simple works better**
  We didn't consider K-mean at the first place, yet we are amazed by its performance in the end. Turned out you don't have to wait hours before random forest is done training to get satisfied result. Always try something simple first.

References

[1] C.-C. Jay Kuo and Yueru Chen, “On Data-Driven Saak Transform”.

[2] Kevin P Murphy, “Machine learning: a probabilistic perspective”

[3] Yaser S. Abu-Mostafa, Malik Magdon-Ismail, Hsuan-Tien Lin, “Learning From Data”