

API for Text Identification

1. Structures

```
#define TYPE_TOTAL_CLASS      3
#define MRF_SIZE_LEVEL       2

#define TYPE_PRINT_TEXT      1
#define TYPE_HANDWRITING     2
#define TYPE_NOISE           3

struct {
    int x;           //x, y, w, h indicate the bounding box of the word zone
    int y;
    int w;
    int h;
    int nType;       //Classification result
    double nConf;     //Classification confidence for the first choice
    double nClassConf[TYPE_TOTAL_CLASS];
                    //Classification confidence for all choices
    int nIndex;      //Reserved
} WORD_ZONE;

typedef struct {
    int nDimension;   //Feature dimension
    int nClassNum;     //Number of classes
    int *nClassLabel; //All class labels
    double **Eff;      //Projection coefficients of the Fisher classifier
    double *nMean1;    //Projection mean for class 1
    double *nVar1;     //Projection variance for class 1
    double *nMean2;    //Projection mean for class 2
    double *nVar2;     //Projection variance for class 2
    int *nClassLabel1; //Label for class 1
    int *nClassLabel2; //Label for class 2
    double *nTH;       //Threshold
}CLASSIFIER;
//Each classifier is trained for a two-class problem. Therefore, M=nClassNum*(nClassNum-1)/2 classifiers should be trained. The dimension of Eff is M*nDimension. The dimensions of nMean1, nVar1, nMean2, nVar2, nClassLabel1, nClassLabel2 are M.

typedef struct {
    int nZone[MRF_SIZE_LEVEL]; //Number of zones for each size level
    int nLabel[MRF_SIZE_LEVEL][TYPE_TOTAL_CLASS];
                    //Number of zones for each class and each size level

    //Clique frequency for Cp
    int ****nPatternP;
    //Clique frequency for Cn
    int *****nPatternN;
    int nCliqueType; //Clique type
    int nClassLabel[TYPE_TOTAL_CLASS]; //Class labels
    int nClassNum; //Number of classes
} MRF_MODEL;
```

2. API

2.1 Layer Separation

```
class CTILayerSeparate{
private:
    WORD *m_pWord; //Segmented words
    int m_nWord; //Number of words
    CLASSIFIER m_Classifier; //Classifier
    MRF_MODEL m_MRF_model //MRF model for post-processing
public:
```

```

        CTILayerSeparate(void);
        ~CTILayerSeparate(void);
public:
    //Set image for processing
    int TISetImage(char *fnImg);
    //Set image from memory for processing
    int TISetImage(BYTE *pImgData, int w, int h);
    //Set MRF model for post-processing
    int TISetMRFModel(char *fnMRFModel);
    //Set classifier
    int TISetClassifier( char *fnClassifier );
    //Word segmentation
    int TIWordSegmentation( );
    //performing word classification
    int TIClassification( );
    //MRF-based post processing
    int TIMRFPPostProcessing( )
    //This function provides a way for users to use his/her own
    //module to perform segmentation, classification of logo, figure, etc.,
    //and then fill information to the structure of WORD.
    int TISetWord(WORD *pWord, int nWord);
    //Similar to the previous function, but the custom's result is
    //stored in ZONE file.
    int TISetWord(char *fnCustom);
    //Get the segmentation and classification results
    int TIGetWord(WORD_ZONE *pWord, int &nWord );

    //Output result to ZONE file
    int TIOutputZone( char *fnZone );
    //Get layer image. nType = 1 -- Printed text, 2 -- Handwriting, 3 -- Noise
    int TIGetLayer( BYTE *pImgData, int &w, int &h, int nType );
    //Save layer to an image file
    int TISaveLayer( char *fnLayerImg, int nType );
};

```

2.2 Train the classifier

```

class CTITrainClassifier
{
private:
    char m_fnTrainClassifierConfig[MAX_PATH];    //Configure file for the
training set
    CLASSIFIER m_Classifier;    //Classifier
public:
    CTITrainClassifier(void);
    ~CTITrainClassifier(void);
public:
    //Set the configure file of the training set
    int TISetTrainConfig( char *fnTrainConfig );
    //Training the classifier
    int TITrainClassifier( );
    //Output trained classifier
    int TIOutputClassifier( char *fnClassifier );
};

```

2.3 Train the MRF model

```
class CTITrainMRFModel
{
private:
    char    m_fnTrainMRFConfig[MAX_PATH]; //Configure file of the training set
    MRF_MODEL m_MRF_Model;    //Trained MRF model
public:
    CTITrainMRFModel(void);
    ~CTITrainMRFModel(void);
public:
    int TISetTrainConfig( char *fnTrainConfig ); //Set configure file
    int TITrainMRFModel( );    //Training the MRF model
    int TIOutputMRFModel( char *fnModel ); //Output trained MRF model
};
```

3. Data Files

There are several data files needed or outputted by the API.

3.1 ZONE File

The default suffix is “.ZONE”. It has the following format

Total Zones: xxx

x, y, w, h, nType

.....

NOTE: nType takes the value of 1 (printed text), 2 (handwriting), or 3 (noise)

Example:

Total Zones: 4

```
1001 1583 94 32 1
1001 2081 193 31 2
1005 1880 66 30 1
1031 580 70 26 3
```

3.2 Training Configure File for Classifier

It has the following format:

```
GROUNDTRUTH_DIR=   Directory of the ground truth
IMG_DIR            =   Directory of the image
GT_SUFFIX          =   Suffix of the ground truth file
IMG_SUFFIX         =   Suffix of the image file
CLASSES           =   Types needed to be classified
```

File1

File2

File3

.....

NOTE: The text line with “----” is demanded to separate the configure and the training file list.

Example:

```
GROUNDTRUTH_DIR= C:\Work\HandPrint\GroundTruth\Word\  
IMG_DIR          = C:\Work\HandPrint\GroundTruth\Word\  
GT_SUFFIX        = .ZONE  
IMG_SUFFIX       = .TIF  
CLASSES          = PRINT_TEXT, HANDWRITING, NOISE
```

```
-----  
0000566613  
0000569324  
0000569672  
0000569705  
0000570581
```

3.3 Trained Classifier File

The format is:

Type=4 Class=NumberOfClass

Fisher Efficient:

xxx xxx xxx

TH=xxx

Class1=x Mean1=xxx Delta1=xxx

Class2=x Mean2=xxx Delta2=xxx

Fisher Efficient:

xxx xxx xxx

TH=xxx

Class1=x Mean1=xxx Delta1=xxx

Class2=x Mean2=xxx Delta2=xxx

.....

NOTE: The number of classifiers are $M = (\text{NumberOfClass} - 1) * \text{NumberOfClass} / 2$. This is the training result of the classifier. Do not try to change it manually.

3.4 Training Configure File for MRF Model

```
GROUNDTRUTH_DIR    = Directory of the ground truth  
GT_SUFFIX          = Suffix of the ground truth
```

```
-----  
File1
```

File2

.....

NOTE: The text line with “----” is demanded to separate the configure and the training file list.

3.5 Trained MRF Model File

Clique Cp

Total xxx zones:

Size 0, Num=xxx (0.xx)

Label 0:xxx (0.xxx)

Label 1:xxx (0.xxx)

Label 2:xxx (0.xxx)

Size 1, Num=xxx (0.xxx)

Label 0:xxx (0.xxx)

Label 1:xxx (0.xxx)

Label 2:xxx (0.xxx)

Size 0, Num=xxx (0.xxx)

(left, center, right)

(0, 0, 0) xx

(0, 1, 0) xx

.....

Size 1, Num=xxx (0.xxx)

(left, center, right)

(0, 0, 0) xx

(0, 1, 0) xx

.....

Clique Cn

Total xxx zones:

Size 0, Num=xxx (0.xxx)

Label 0:xxx (0.xxx)

Label 1:xxx (0.xxx)

Label 2:xxx (0.xxx)

Size 1, Num=xxx (0.xxx)

Label 0:xxx (0.xxx)

Label 1:xxx (0.xxx)

Label 2:xxx (0.xxx)

Size 0, Num=xxx (0.xxx)

(left, center, right)

(0, 0, 0, 0, 0) xx

(0, 0, 0, 0, 1) xx

.....

Size 1, Num=xxx (0.xxx)

(left, center, right)

(0, 0, 0, 0, 0) xx

(0, 0, 0, 0, 1) xx

.....

NOTE: This is the training result of the MRF model. Do not try to change it manually.