**HISTOGRAM OF GABOR PHASE PATTERNS: C++ CODE v0.1**

In this document we provide an overview of the code interface of the feature extractor implemented for extracting the Gabor phase Patterns of an input image. This code is an implementation/adaptation of the algorithm presented in [1]. The algorithm proceeds by taking the self-quotient image of an image, followed by computing its response to a bank of Gabor filters. Then the Gabor phase patterns are computed followed by a histogram of gabor phase patterns. The code is implemented using object-oriented programming in C++. It has been written with the objective of making the code modular. Thus in what follows we describe the various classes used in implanting the algorithm. We mainly include the methods associated with the respective classes. Member variables are mostly not mentioned for the sake of brevity.

**Part A: Class Structure**

The four main classes implemented are:

1. FeatureExtractor- Implements the feature extraction algorithm.
2. GaborFilter: Computes the gabor filter bank and its response to a given input image
3. SQI: Implements the SQI algorithm given an input image
4. GaborPhasePatterns: Implements to algorithm to compute the local and global gabor phase patterns
5. HGPP: Implements the algorithm to compute the histogram of gabor phase patterns. It inherits from the base class **GaborPhasePatterns.**

**Class: FeatureExtractor**

class FeatureExtractor {

protected:

cv::Mat inp\_img;

SQI sqi;

HGPP hgpp;

GaborFilter filt;

public:

//constructor

**FeatureExtractor**(…);

//destructor (add later)

**~FeatureExtractor();**

// reset some structures after extraction features from an image

**void ResetStructures();**

//Display Image

**void DisplayImage(cv::Mat img) const;**

// add new image for this extraction method

**void AddNewImage(cv::Mat img);**

// set parameters

**void SetParameter(std::string name, std::string value);**

//Extract HGPP Feature(the sub-classes are used in this method)

**void ExtractHGPPFeature();**

//Retrieve relevant output structure

**Mat RetrieveOutputFeature(bool isSQI,bool isReal,bool isGGPP);**

};

**Class: GaborFilter**

class GaborFilter{

public:

//constructor,destructor

**GaborFilter(..);**

**~GaborFilter();**

//Computational Methods

**void ComputeFilterResponse(cv::Mat);**

**void ComputeFilter();**

//Interface

**void SetParameter(std::string name,std::string value);**

**vector<cv::Mat> GetFilterOutput(bool isReal) const;**

**void PrintParameters()const**

private:

//Member variables

Mat inp\_img;

std::vector<cv::Mat> GaborFilt\_r,GaborFilt\_i;

std::vector<cv::Mat> FiltResp\_r,FiltResp\_i;

//Internal method to reset I/O structures upon change of parameters

**void ResetStructures();**

};

**Class SQI**

class SQI

{

public:

//constructors

**SQI();**

**~SQI();**

//Accesors

**void setInpImage(cv::Mat img);**

**void setParameters(int imsiz\_,int size\_,float sigma\_);**

**cv::Mat getOutImage()const;**

//Display SQI Image

**void DisplayImage() const;**

//Main method to get sqi image (includes some normalization as pre-processing)

**void get\_self\_quo\_image();**

private:

**cv::Mat in\_img;**

**cv::Mat out\_sqi;**

};

**Class: GaborPhasePatterns**

class GaborPhasePatterns{

public:

// constructors/destructors

**GaborPhasePatterns(…);**

**~GaborPhasePatterns();**

//Accesors/Interface methods

**void SetParameter(int \_scale,int \_orient**);

**void PrintParameters()const;**

**void setFilteredInput(vector <cv::Mat>Inp);**

**vector<cv::Mat> GetGPP(bool isReal,bool isGGPP);**

//Computational methods

**void ComputeLGPP(bool isReal);**

**void ComputeGGPP(bool isReal);**

protected:

**vector<cv::Mat> LGPP\_r,LGPP\_i;**

**vector<cv::Mat> GGPP\_r,GGPP\_i;**

**vector<cv::Mat> Resp;**

// the gabor filter response used as input to the //ComputeLGPP/ComputeGGPP methods

};

**Class: HGPP (inherits from GaborPhasePatterns)**

class HGPP : public GaborPhasePatterns

{

public:

//constructors

**HGPP();**

**~HGPP();**

//Accesors

**void SetParameter();**

**void PrintParameters() const;**

**void PrintHistogramStats()const;**

**Mat GetOutputHistogram(bool isReal,bool isGGPP) const;**

//Main Computation Method

**void ComputeHistogram(bool isReal,bool isGGPP);**

protected:

Mat HistGPP[4];

int NoLGPP,NoGGPP;

**int ComputeNoBlocks(bool isReal,bool isGGPP);**

};

**Part B: User-controlled options**

We mainly use the method **FeatureExtractor::SetParameter()** to set the relevant parameters for the feature extraction method. These parameters are

|  |  |  |  |
| --- | --- | --- | --- |
|  | Parameter | Description | Default Value |
| 1 | scale | Number of scales in Gabor Filter Bank | 5 |
| 2 | Orient | Number of orientations in Gabor Filter Bank | 8 |
| 3 | NBINS | # of bins per image-patch in histogram | 16 |
| 4 | size | Size of blurring kernel in Self-Quotient Image(SQI) | 9 (i.e. 9x9) |
| 5 | sigma | Variance of blur kernel in SQI | 3.0 |
| 6 | IMSIZE | Size of input image | 64 |
| 7 | FILTSIZ | Size of Gabor filter kernel | 65 |
| 8 | fmax | Frequency of gabor filter kernels | 0.5\*pi |
| 9 | sigma\_g | Spatial aspect ratio of gabor filter kernels | 2\*pi |
| 10 | region\_size\_x | Size of image patch for histogram (horizontal direction) | 8 |
| 11 | region\_size\_y | Size of image patch for histogram (vertical direction) | 8 |
| 12 | overlap\_x | Overlap of image patches for histogram (horizontal) | 4 |
| 13 | overlap\_y | Overlap of image patches for histogram (vertical) | 4 |

|  |
| --- |
|  |

**Part C: Simple Example**

We show how to use classes defined above to write a simple program for feature extraction from images.

**int** **main**(**int** argc, **char\*** argv[]) {

   Mat img;

   FeatureExtractor feat;

   cout**<<**"Setting Input Parameters...\n";

*// Set input Parameters for the various methods*

feat.SetParameter("scale","5");

feat.SetParameter("orient","8");

       feat.SetParameter("NBINS","16");

       feat.SetParameter("size","9");

       feat.SetParameter("sigma","3.0");

       feat.SetParameter("IMSIZE","64");

       feat.SetParameter("FILTSIZE","65");

       feat.SetParameter("fmax",std**::**to\_string(.5**\***pi));

       feat.SetParameter("sigma\_g",std**::**to\_string(2**\***pi));

feat.SetParameter("region\_size\_x","8");

feat.SetParameter("region\_size\_y","8");

feat.SetParameter("overlap\_x","4");

feat.SetParameter("overlap\_y","4");

*//feature extraction*

img **=** cv**::**imread("lena.png"); *// load image*

feat.DisplayImage(img);

  feat.AddNewImage(img);*// add image to extraction*

       cout**<<**"Perform Feature Extraction...\n";

*// Extract Feature from image*

*//In the simplest case this involves taking the self-quotient image*

       feat.ExtractHGPPFeature();

*// save results*

*// Method is definedto save either the Self-quotient image or the histograms of various Gabor Phase Patterns*

**bool** isSQI**=**true,isReal**=**true,isGGPP**=**true;

cout**<<**"Retrieve output feature"**<<**endl;

Mat tmp**=**feat.RetrieveOutputFeature(**!**isSQI,isReal,**!**isGGPP);

string str**=**"outputFeatures.feat";

FileStorage fs(str,FileStorage**::**WRITE);

       fs **<<** "Feature" **<<** tmp;

       fs.release(); *// explicit close*

       cout **<<** "Output Write Done to " **<<** str **<<** endl;

**return** 0;

}

**REFERENCES**

[1] Baochang Zhang; Shiguang Shan; Xilin Chen; Wen Gao, "**Histogram of Gabor Phase Patterns (HGPP): A Novel Object Representation Approach for Face Recognition**," *Image Processing, IEEE Transactions on* , vol.16, no.1, pp.57,68, Jan. 2007  
doi: 10.1109/TIP.2006.884956