FOSSEE Image Processing and Computer Vision Toolbox
Summer Internship 2017, IIT Bombay

Shreyash Sharma

May-July 2017

Acknowledgements

I would like to extend my gratitude to thank Ms. Shamika Mohanan for giving me the opportunity to work as an intern under FOSSEE at IIT Bombay and also for guiding me as a mentor throughout my project. I would also like to thank Prof. Kannan Moudgalya for his valuable inputs and continuous support. I would also like to thank my teammates and colleagues for their continuos contribuiton and support.

Prof. Madhu Belur
CO-PI,FOSSEE
Dept. of Electrical Engineering,IIT Bombay

Ms. Shamika Mohanan

Project Team

- Shreyash Sharma
  Manipal Institute of Technology, Manipal
  Department of Computer Science Engineering

- Gursimar Singh
  PDPM IIITDM, Jabalpur
  Department of Electronics Communication Engineering

- Nihar Rao
  Manipal Institute of Technology, Manipal
  Department of Computer Science Engineering

- Manoj Sree Harsha
  National Institute of Technology, Nagpur
  Department of Computer Science Engineering

- M Avinash Reddy
  IIT Hyderabad
  Department of Electrical Engineering

- Shubham Lohakare
  NITK Surathkal
  Department of Computer Engineering

- Ashish Manatosh Barik
  NIT Rourkela
  Department of Computer Science Engineering

- Rohan Gurve
  G H Patel College of Engineering and Technology, Anand
  Department of Computer Engineering

- Ebey Abraham
  Rajiv Gandhi Institute of Technology, Kottayam, Kerala
  Depatment of Computer Science

- Siddhant Narang
  Manipal Institute of Technology, Manipal
  Department of Computer Science Engineering

Contents

Chapter 1

Introduction

1.1 Aim

The aim of this project was to add new functionalities to the Scilab Image Processing and Computer Vision Toolbox by creating OpenCV and PCL wrapper functions in Scilab and also to port the pre-existing codebase from OpenCV 2.4 to 3.0.0.

The project invovled working with two libararies OpenCV and Point Cloud. OpenCV is an open source image processing computer vision library while Point Cloud is an open source library for manipulating 2D/3D images and point cloud processing.

The functions in this toolbox can be classified into two categories:

- Image Processing
- Computer Vision

1.1.1 Image Processing

This category contains all the functions that deal with the manipulation of images which includes all the image transformation, enhancement and input/output functions.

1.1.2 Computer Vision

This category contains all the functions that deal with detection and recognition of images. Functions in this category can also be used for objct tracking, camera calibration(Normal and Fisheye) and 3-D point cloud processing.

1.2 Scilab

Scilab is an open source toolbox which can be used to solve a variety of problems. The motivation behind working on the Scilab toolbox is the fact that its proprietary counter part Matlab incurs a very high cost of purchasal.

In this project we have used Scilab as a platform to serve the user with the capability to solve problems related to image processing and computer vision.

1.2.1 Scilab API

Scilab[1] like any other toolbox in the market has APIs (Application Program Interface) written for C,C++ and other languages. In this project we have used APIs in C++ to create the wrapper functions for OpenCV and Point Cloud Library(PCL).

1.3 OpenCV

OpenCV[2] is an open source library for image processing and computer vision which was written for langueages like C/C++, Python and Java.

In this project we have used its functions in C++ as they could be easily integrated with the Scilab APIs to create the wrapper functions.

OpenCV is highly customizable as further in this report you will see we modified the OpenCV source code to serve our purpose.

1.4 Point Cloud Library (PCL)

The PCL[3] library like OpenCV is also an open source library which is specifically used for manipulating 2D/3D image and point cloud processing.

It also has its APIs for C++ which we have used to create the wrapper functions for a corresponding PCL functions.

Chapter 2

Interface Design

2.1 Interface Structure

The following things constitute the important parts of the interface.

1. builder.sce
2. loader.sce
3. C++ files
4. Scilab Macro Files(.sci files).

2.1.1 builder.sce

This file builds the required functions and prepares them so that they can be loaded into the Scilab environment. It contains the following:

- The list of C++ files that have to be compiled
- The functions to be loaded in the Scilab environment.
- The linker flags
- The macros to be loaded
- Code to build the documentation, help files and demos

2.1.2 loader.sce

This file is used for loading the files which have been built by executing the builder file.

2.1.3 C++ Files

These are the files used for building the wrapper functions executable in scilab.It functions as the backend for the scilab functions.The main problem solving occurs within these files itself.

2.1.4 Scilab Macro Files

Instead of interfacing with the Scilab command line directly, the C++ functions are called by Scilab functions defined in the macro files. The macro files help in building of the scilab functions by providing the basic skeleton and calling the c++ function within.

Macros serve the following functions:

- Validating the user input data and provding error checks for set of inputs.
- Proper formatting of the data for different functions in scilab.
- Support variable input and output arguments with ease

Chapter 3

Implementation of OpenCV functions

3.1 Machine Lerning

This module[4] contains submodules for training and classfication of the data presented by the user.The functions present, train the data on imagesets and store them in an .xml or .yml files. The file is then used for classification of data based on whether the training type is supervised or unsupervised.Some work was previously done on this module, we built upon that work to add new functionalities and make it more user friendly.

- Artificial Neural Networks and Multi layer[8] perceptron(ANNMLP)
- Decision Trees(dtrees)[9]
- Boost(Boost)[10]

3.1.1 Artificial Neural Networks and Multi layer perceptron

The Multi Layer Perceptron (MLP) algorithm is a powerful form of an Artificial Neural Network that is commonly used for regression (and can also be used for classification). The Multi Layer Perceptron algorithm follows supervised learning algorithm which can be used for both classification as well as regression for various types of N-dimensional signals .

trainANNMLPClassifier

MLP follows supervised learning.This function trains a ANN classifier which can be used to predict classes of images given to it as input using the predictANNMLP() function.

The function can exploit two types of algorithims which include back propagation and resilient backpropagation. The algorithm stops when either 100 iterations are completed or the error rate is ¡ 107.

predictANN

The predict fucntion takes in input from the output of the train function and a test image for predicting the label of the input image.

3.1.2 Decision Trees

Decision trees ,as predictive models, are used for making observations about items which are represented as branches of the tree as well as for concluding about the item's target value which are represented by leaves. The dtree models having target variables in the form of discrete values are termed classification trees, in such kind of tree structures, leaves are represented by class labels while branches are used to represent conjunctions of features.

traindtreeClassifier

Dtree follows supervised learning.This function trains a Dtree classifier which can be used to predict classes of images given to it as input using the predictdtree() function.

The algorithm stops when either 100 iterations are completed or the error rate is ¡ 107.

predictdtree

The predict fucntion takes in input from the output of the train function and a test image for predicting the label of the input image.

3.1.3 BoostClassifier

Boosting in general is an ensemble method that creates a strong classifier from a given number of weak classifiers. This is done by building a model from the training data, then creating a second model that attempts to correct the errors from the first model.

trainBoostClassifier

Boost follows supervised learning.This function trains a Boost classifier which can be used to predict classes of images given to it as input using the predictBoost() function.

The algorithm stops when either 100 iterations are completed or the error rate is ¡ 107.

predictBoost

The predict fucntion takes in input from the output of the train function and a test image for predicting the label of the input image.

3.2 Object Detection

This module[6] comes under the subcategory of Computer Vision of our toolbox.It lists all the functions that can be used to detect objects.

- Histogram of Oriented Gradients[11](HOG)
- Local Binary Pattern[12](LBP)

3.2.1 extractHOGFeatures

This function is used to extract features from an image from histogram of gradients and store the features in the form of a matrix and return this matrix to the user.This helps the user in using the output feature matrix further in their call sequence.

3.2.2 extractLBPFeatures

This function is used to extract features from an image by comparing pixel values between the central and neighbouring pixels,further , it stores the features in the form of a matrix and returns this matrix to the user.This helps the user in using the output feature matrix further in their call sequence.

These functions were implemented beforehand.The extractHOGFeatures had to be changed syntactically to make it running on the new stable version of OpenCV, while extractLBPFeatures involved changing it's LBP histogram calculation function.

3.3 2D Features Framework

This module[5] comes under the subcategory of Computer Vision of our toolbox.It contains all the functions that are used to detect,extract,match nad draw common features of 1 or more images.

- Agast Features(AGAST)[10]
- Accelerated-Kaze Features(AKAZE)[11]

3.3.1 detectAGASTeatures

This function is used to detect corner Features in an input image.It uses AGAST algorithm to detect the features. The function is part of the calling sequnence which includes feature detection and drawing of keypoints.

3.3.2 detectAKAZEFeatures

This function is used to detect and Compute Features in an input image.This function uses accelerated-KAZE algorithm to detect and compute the features. The function is part of the calling sequnence which includes feature detection, extraction, matching of features and subsequenlty drawing them.

3.4 Camera Calibration

This module[7] is a part of Computer Vision of our toolbox.It contains all the functions that are required for camera calibration and 3D reconstruction.

3.4.1 Calibrate

This function[13] is used for undistortion of an image using camera calibration. The Camera involved belongs to fisheye/pinhole model.The intrinsic and extrinsic parameters of the camera involved are computed.

The Following functions were already implemented beforehand, They were required to be ported from openCV 2.4 to 3.0.0. This also involved some minor bug fixes.

- distortPoints
- undistortImage

3.5 Other Functions

The Following functions have been implemented were already done beforehand,but required some bug fixes and logical corrections.

3.5.1 imhistmatch

The logic had to be modified for normalizing the hostograms of the input images

3.5.2 epipolrlines

The logic had to be modified completely for making the function fit in the already present calling sequence.

3.5.3 isEpipoleinImage

The logic had to modified to make the function accept stereo images for detection of epipolelines.

These functions had to be ported to the current stable OpenCV version.

- ellipse2poly
- geometricshearer
- imfuse
- imgaussfilt3
- imguidedfilter
- imhmax

- imsharpen
- lab2uint16
- lab2xyz
- puttext
- rgb2ntsc
- sobel
- threshold
- projectpoints
- estimatenewcameramatrixfromstereorectify
- impixel
- mean1
- multithresh
- dctmtx

Chapter 4

Implementation of Point Cloud Library Functions

This module involves two functions which were implemented using point cloud library for scilab.

- pcread
- PointCloud

4.0.1 pcread

This function is used to read a .ply or a .pcd file and return the parameters calculated in the form of a scalar or matrix.

4.0.2 PointCloud

This function is used to convert the input parameters obtained from pcread to a structure.

Chapter 5

Conclusion and Future Work

The work throughout the period was mainly focused on the use of the OpenCV library to develop the image processing toolbox for Scilab. Some contribution was also made using the of point cloud library,But there is a lot of scope for working on pcl library for the future.

Bibliography

[1] https://help.scilab.org/docs/6.0.0/en_US/section_204636e951f595409bc6782bb8e1d2d9.html

[2] http://docs.opencv.org/3.0.0/

[3] http://pointclouds.org/

[4] http://docs.opencv.org/3.0.0/dd/ded/group__ml.html

[5] http://docs.opencv.org/3.0.0/da/d9b/group__features2d.html

[6] http://docs.opencv.org/3.0.0/d5/d54/group__objdetect.html

[7] http://docs.opencv.org/3.0.0/d9/d0c/group__calib3d.html

[8] http://docs.opencv.org/trunk/d0/dce/classcv_1_1ml_1_1ANN_MLP.html

[9] http://docs.opencv.org/trunk/d8/d89/classcv_1_1_ml_1_1DTrees.html

[10] http://docs.opencv.org/3.1.0/d6/d7a/classcv_1_1ml_1_1Boost.html

[11] http://docs.opencv.org/trunk/d7/d19/classcv_1_1AgastFeatureDetector.html

[12] http://docs.opencv.org/3.0-beta/doc/tutorials/features2d/akaze_matching/akaze_matching.html

[13] hhttp://www.learnopencv.com/histogram-of-oriented-gradients/

[14] http://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/

[15] http://docs.opencv.org/trunk/db/d58/group_calib3d_fisheye.html