

Bug Reporting and Functional Analysis of Packages in Octave

A Project Report Submitted in the partial fulfillment of the requirements for
the award of the degree of

Bachelor of Technology in Department of Electronics and Communication Engineering

By

M. Syamanth	(Roll No 150040531)
V. Tharun Kumar	(Roll No 150040923)
P. T. V.V. Sai Ram	(Roll No 150040701)
V. Akhil Vamsi	(Roll No 150040964)

Under the Supervision of
Mr. K. Sripath Roy
Assistant Professor,
Department of E.C.E



KONERU LAKSHMAIAH EDUCATION FOUNDATION,

Green Fields, Vaddeswaram-
522502, Guntur(Dist), Andhra
Pradesh, India.

November - 2018

**KONERU LAKSHMAIAH EDUCATION FOUNDATION
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**



Declaration

The Project Report entitled “**Bug Reporting and Functional Analysis of Packages in Octave**“ is a record of bonafide work of M. Syamanth (150040531), V. Tharun Kumar (150040923), P. T. V. V. Sai Ram (150040701), V. Akhil Vamsi (150040964) submitted in partial fulfillment for the award of Bachelor of Technology in Electronics and Communication Engineering during the academic year 2018-19.

We also declare that this report is of our own effort and it has not been submitted to any other university for the award of any degree.

M. Syamanth	(150040531)
V. Tharun Kumar	(150040923)
P. T. V. V. Sai Ram	(150040701)
V. Akhil Vamsi	(150040964)

KONERU LAKSHMAIAH EDUCATION FOUNDATION

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING



Certificate

This is to certify that the Project Report entitled “**Bug Reporting and Functional Analysis of Packages in Octave**” is being submitted by M. Syamanth (150040531), V. Tharun Kumar (150040923), P. T. V. V. Sai Ram (150040701), V. Akhil Vamsi (150040964) in partial fulfillment for the award of Bachelor of Technology in Electronics and Communication Engineering during the academic year 2018-19.

Signature of the Supervisor

Mr. K. Sripath Roy
Asst Professor

Signature of the HOD

Dr. K. Ch. Sri Kavya
Head of the Department

Signature of the EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our project supervisor **Mr. K. Sripath Roy, Assistant Professor** for his time and his valuable advice and guidance throughout our project. Without his support and encouragement we would not have completed our work.

We also express our kind gratitude to our Head of the Department **Dr. K. Ch. Sri Kavya** for her constant support and encouragement throughout the semester. The care and concern she has shown for us help us overcome all the problems and obstacles we have faced throughout the project work.

We also wish to say a sincere thank you to the Open Source community and **KLUGLUG** for their support and for helping us to develop our project. We will be forever grateful to them

M. Syamanth (150040531)

V. Tharun Kumar (150040923)

P. T. V. V. Sai Ram (150040701)

V. Akhil Vamsi (150040964)

ABSTRACT

Free end open source technology is one of the platforms to work on open source tools which are available to everyone. Unlike some other tools the open source tools are available to every section of the community. The contribution to the open source tools involves bug fixing, developing new packages, modifying new packages etc., One of the open source tool we are working on is Octave which we are doing is package contribution and bug fixes. We are developing a function and implementing it in Octave.

TABLE OF CONTENTS

CHAPTER	TITLE OF THE CONTENT	PAGE NO
	Abstract	v
1	INTRODUCTION	1-4
	1.1 What is Free Software	1
	1.2 Whom does it benefit?	1
	1.3 Is Open Source for free?	2
	1.4 Advantages and Disadvantages	3
	1.5 Comparing with Closed Source softwares	3
	1.6 Why do we use free software?	4
2	LITERATURE SURVEY	5-19
	2.1 What is Octave	5
	2.2 Octave Forge	5
	2.2.1 Package installation	6
	2.2.2 Using Packages	9
	2.3 Creating a package	9
	2.4 Adding and Removing Packages	14
	2.5 History behind IRC	18
3	Free Software Contribution	20-40
	3.1 Bug Fixing	20
	3.2 Package Contribution	28
4	Conclusion and Future Scope	40
5	Future Scope	41

LIST OF FIGURES

Figure	Name of the Figure	Page no
2.1	Bug reports at Octave forge	7
2.2	Select search for already reported bugs	7
2.3	Search for the name of the package to identify bugs	8
2.4	Selecting a bug to solve	8
3.1	Reporting the bugs in symbolic package	26
3.2.1	Showing the repository in octave-forge	29
3.2.2	Showing the repository and forking method	29
3.2.3	Cloning the package	30
3.2.4	Input 1 of imadjust()	31
3.2.5	Output 1 of imadjust()	32
3.2.6	Output 1 pixel values of imadjust()	33
3.2.7	Input 2 of imadjust()	33
3.2.8	Output 2 of imadjust()	34
3.2.9	Input 1 of imnoise()	35
3.2.10	Output 1 of imnoise()	35
3.2.11	Input 1 of ROI	36
3.2.12	Output 1 of ROI	37
3.2.13	Input 1 of bwmorph()	38
3.2.14	Output 1 of bwmorph()	39
3.2.15	Output 1 of bwmorph()	39

CHAPTER-1

INTRODUCTION

1.1 What is Free Software

Free software refers to the software development that anybody can use, modify or enhance. We can manipulate the source code that is present and we can modify the program and we can decide which we can use and we we need not. The programmers can fix the parts of the code that do not work correctly and improving them for adding for better results. Only the original originators of the code will have the access to copy, inspect or alter the software. In order to use the software, we must essentially sign a license displayed the first time they run it that they should work under the terms and policies of the software agreement and they don't have the freedom to go beyond.

Free software authors make their source code visible to others who want to see their code. They can even copy, learn from it, alter it or change it. However the legal terms differ from proprietary terms. The licenses considerably effect the way we alter, enhance and distribute the code. In general, the users will have free access to do whatever they want with the code that was provided for them. Some licenses stipulate that anyone who alters or change a code must share the program without charging license fee for it.

1.2 Whom does it benefit?

Open Source Technology benefits both computer programmers and non programmers. In early days tools like Linux Operating Systems and Apache Web Server applications are both open source tools and internet is built much from these open source tools. Every time we use a email or chat with friends or play

Introduction

video games or use mobile phones they all connect to a console network of computers that uses open source tools and software that connects the devices.

People also prefer to use open source tools for various benefits:

1. **Control:** Most people prefer to use open source tools because they can instantly know what's going to happen in front of them and they can change the parts if they don't like them.
2. **Training:** Most people also use open source tool to become a better programmers and as they are freely accessible to students they can easily learn from them and they also gain ample knowledge. They can also share their work with others inviting them to be a part of their own projects and helping them solving their problems and mistakes.
3. **Security:** Some people also prefer open source tools because they consider it more stable and secure. Anyone can correct the mistakes they have done and the mistakes can be modified by anyone who has access to their files. If anyone found any weak links that can be exposed they can rectify them because it is accessible to everyone.

1.3 Is Free software for free?

Open source is free but not in all cases. The authors can charge the money for the program they are sharing or contributing. But when they have to sell the software to others, the programmers earn huge profits by selling them. The programmers only provide support and not necessarily the entire source code. The software remains free and they earn money by helping others with their installation problems or troubleshooting problems.

1.4 Advantages and Disadvantages:

The Open Source is easier to available than any other proprietary software. Open Source tools can increase the standards and have a sense of ownership to the product. Open Source tools help promote company's image including it's products. They offer flexibility and innovation for rapid growth and development. It is more reliable and more secure because thousands of developers are working on the Open Source tools to identify the bud fixes and issues. It is not dependent on any other company or software.

However, not all open source tools are successful. Some tools lack quality and it is extremely difficult to produce a tool that have a wide commercial applications in the market with some Open Source tools. Some technical requirements may be satisfied but market requirements may not be satisfied. Moreover Open Source allows hackers to identify their loopholes and allows them to expose them thus enhancing some risk in using them. But not all tools are under protective. Some tools are very well upgraded

1.5 Comparing with Closed Source softwares:

The Closed Source software methods have various constraints on how to use the software and what should be done in the code. But the Free End Open Source methods does not limit the use of the software. The source code is given away to other users and it can be modified. The software can be distributed for free as long as credit is given to the manufacturer.

Innovation is another major aspect in the Free end open source softwares. The proprietary models have licensed versions that always do not give away the source code to other developers. However Open Source softwares provide free access to every developer and any developer with better ideas will randomly develop the program and helps to its growth.

Code analysis also varies in free end open source tools as they are even more secure and stable than closed source software programs. Closed software

programs are developing every minute by developers across the world thus having better code quality and better results than closed software sources.

1.6 Why do we use free software:

Free software is one of the basic tool to develop and the ethical choice for others to learn new things and gain knowledge without any fuss. It can be used as a foundation to learn for the society as well as the individuals as the free software tools can be used by everyone.

Many people around the world use the proprietary softwares that requires various licenses and it may or may not be available to everyone. There is a very good chance that we may suffer punishment if we hand over a copy or two violating the terms and conditions of the license agreement. We accept the license agreements which we do not even read and we face the consequences later.

But free software programs are having no license documents and we can use it freely and we can install and remove any time we want irrespective of our usage

CHAPTER-2

LITERATURE SURVEY

2.1 What is Octave?:

GNU Octave is a high level programming language for numerical interpretations and solving linear and non linear problems. GNU Octave is a perfect alternative to Matlab. It was first developed in 1988 by John W. Eaton. It is written in standard C++ language. Octave includes Graphical User Interface (GUI) along with Command line Interface (CLI). Octave is made free of use and is available in Windows, macOS, Unix and Unix like operating systems.

We can install Octave in Linux by typing **sudo apt-get install octave** in terminal. It will automatically get installed and there will be no inbuilt packages present in Octave. We have to install each and every package if we want to use them. The latest version of Octave available is Octave 4.2.2.

If we have to run the Octave we have to type octave in terminal or directly double click octave folder in the pc. In case if you have any trouble using octave you can always type Control-c, it will automatically return to octave prompt. To exit octave you have to type quit or exit in the octave prompt.

Octave also has some inbuilt data types like real and complex scalars, matrices, character strings, a data structure type, and array of all data types. We can also develop new specialized data types by writing a small C++ code.

2.2 Octave Forge:

If we want to contribute to existing package by writing some -m files or a package we want for development then Octave Forge is the place. Octave Forge is essentially a list of inbuilt packages in Octave. We can install packages of our requirements from the octave forge.

2.2.1 : Package installation:

We can find the list of packages in Octave forge by clicking on packages link at the top.

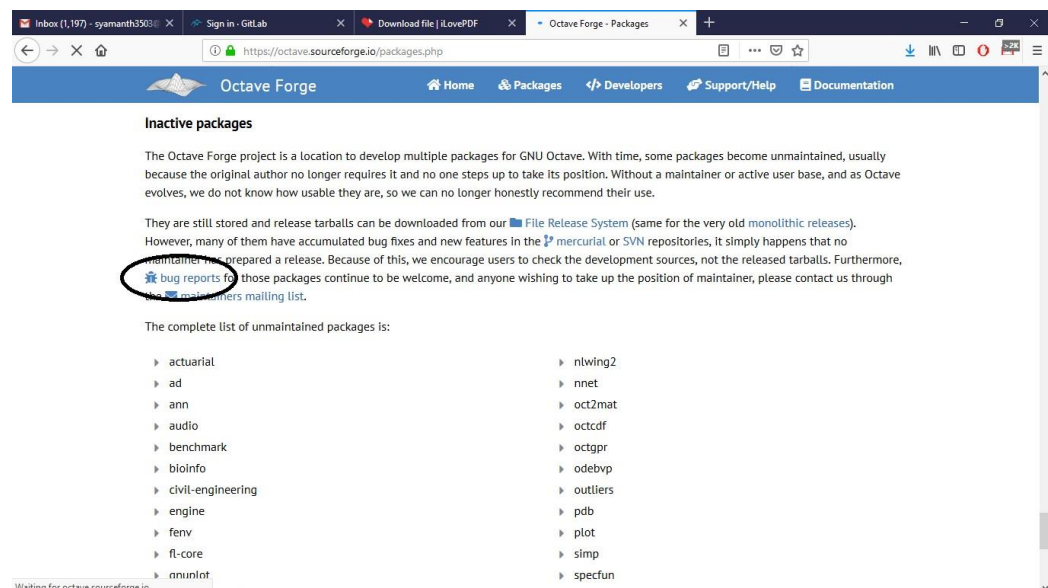
To install a package type **pkg install -forge package_name** in octave prompt.

Octave has Community packages and External packages. Some community packages are Arduino, Communications, Symbolic, Data smoothing, dataframe, gsl, image etc.,

Some External packages are bim, bsplt, divand, fits, nan, ocs etc., It also has Unmaintained packages present. These packages are unmaintained because the developer may no longer require those packages. Those are stored in the website and it is open for bug fixes.

GNU Octave uses bug tracker at Savannah. It is a platform for finding new bug, report a new bug.

The below image shows us the location of **bug reports** in Octave Forge



Introduction

Fig 2.1 Bug Reports at Octave Forge

Then select **search for already reported bugs at the bug tracker**

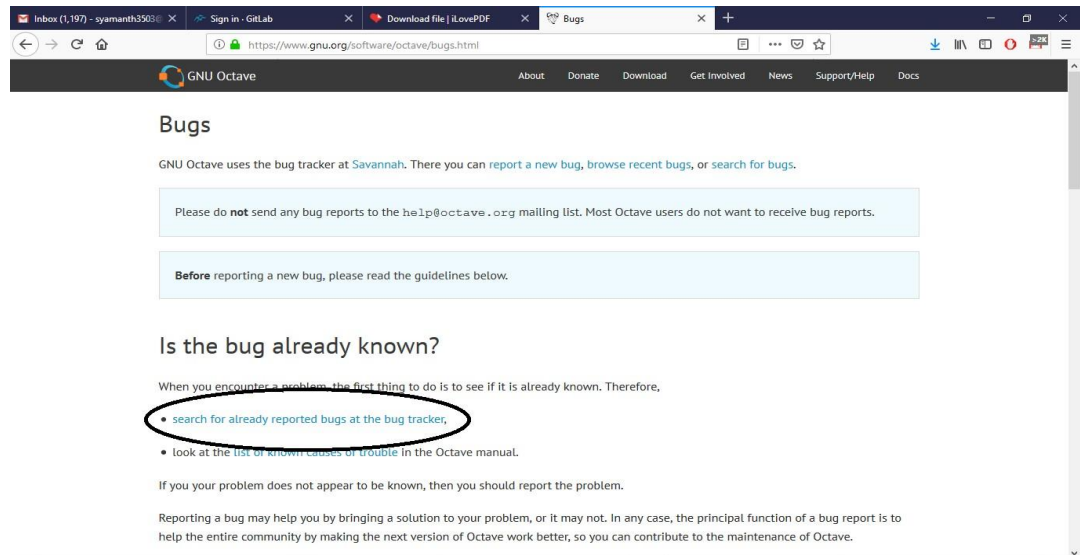


Fig 2.2 Select search for already reported bugs

Then choose the name of the package in which we want to find the bug to fix. Type the package name in the search box and select bugs in the select column and press search

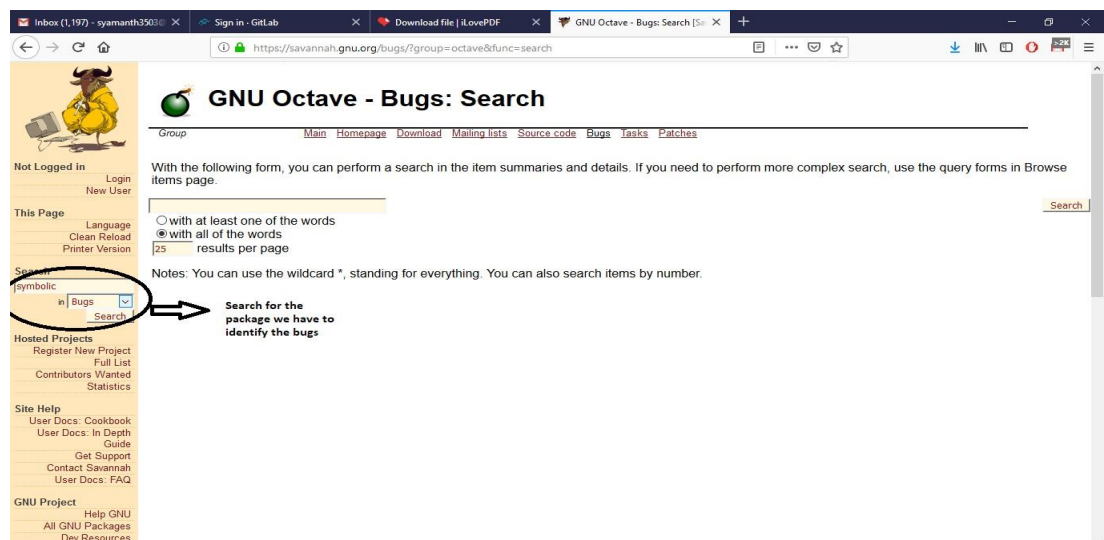


Fig 2.3 Search for the name of the package to identify bugs

Introduction

Then various bugs will be shown to us in the page. Select any one bug you want to fix and we can upload it back to the developer

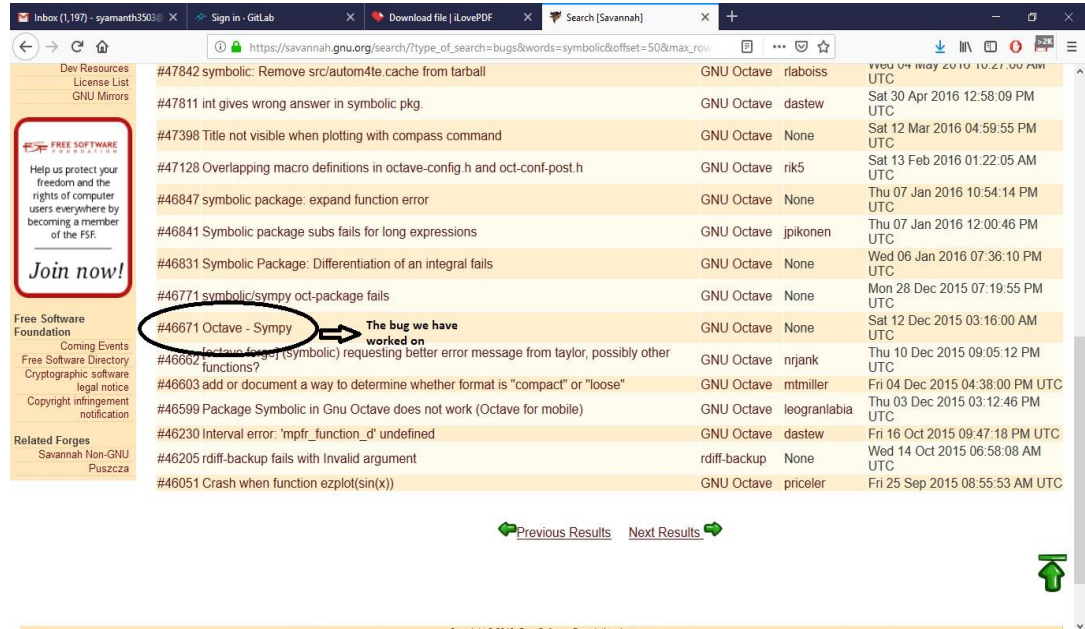


Fig 2.4 Selecting a bug to solve

2.2.2 Using Packages:

By default no packages are present in Octave. We have to install the package in Octave prompt using the command **pkg install -forge package_name**.

Once we have installed we should be able to use the package we have installed. The packages are added to Octave path by typing **pkg load package_name**. In the same way we can unpath by typing **pkg unload package_name**.

2.3 Creating a package:

Basically a package is a compressed zip folder that has basic inbuilt functions. The package may contain the following files:

Introduction

✓ **package/citation:**

This file tells us how to cite the package for publication. Once we type the function citation in the prompt it will display the details

✓ **package/copying:**

This file contains the license of the package. If the license has some linked functions it should be compatible with the general license.

✓ **Package/Description**

The description file contains the information about the package like name of the author, package name, version etc.,

The format of the Description file is:

Name: Name of the packages

Version: x.x.x(Sample version)

Author: Name of the author

Maintainer: The name of the current package maintainer

Title: Title of the packages

Description: A short description of the package

License: The licensed file of the package.

The package manager basically recognizes some of the following keywords:

Name

Name keywords indicates the product manager the name of the package which the user is currently using

Introduction

Version

The version keyword indicates the version of the package which we are using and he checks if there are any updates required to update the current packages

Date

The date keyword tells us about the date of the package that was last updated. If the package has any new updates the manager will know it by typing the date keyword in the command prompt

Author

Author keyword is used to tell us the original author of the package. The author and the package manager may not be same. The package manager is the current owner of the package and the author is the permanent owner of the package.

Maintainer

Maintainer keyword tells us the current maintainer of the package. The maintainer works under the package manager and the packager manager will control the operations of the maintainer.

Title

Title keyword is a one line description of the package. The title and the name of the package are different. The title gives us the brief idea of the package and name of the package is just the name of the package.

Description

Description is a one paragraph description of the package. The description keyword tells us the indepth idea and analysis of the package. It tells

Introduction

us the functioning of the package and the main keywords and functions that are used in the package.

Categories

Categories keyword tells us the detail description of the package. If Index file is not given in the package the categories file is mandatory.

Problems

Problems keyword gives the users the list of problems that are present in the package. It is then the users choice to solve those problems and contribute to the package or tell the maintainer of the package about the problems and help modify those problems.

URL

It gives us the list of URL's that are linked with the package.

Depends

Depends keyword gives us the list of other octave packages that the package is dependent on. Some Octave packages are dependent on other package like for example communications package in octave is dependent on signal package. If the signal package is not installed then we cannot use the communications package. This is called dependency.

License

File describing about the license of the package. The license file is not mandatory as copying file is mandatory.

SystemRequirements

Systemrequirements gives us the external compatibility the package is dependent on. The system requirements are not checked by the package manager and they are to be checked by the user.

BuildRequires

The Build requires keyword gives us the external build dependencies of the package. The system requires keyword gives us the external install dependencies of the package. The buildrequires and systemrequires are not checked by the package manager. They are to be checked by the user.

If the developer wants to add some more arguments he is free to add if he wants to.

✓ **Package/channel log:**

This file gives us the changes that can be made or already made in the package.

The channel log gives us the changes that happened to the package recently.

✓ **Package/index:**

This index file gives us all the functions that are already present in the package. The index file is created automatically from the functions of the package and categories keyword in description file.

✓ **Package/NEWS:**

This file describes all the changes that the user has made to the package that are worth mentioning. The changes that get old due time all moved to package/onevs.

Introduction

✓ **Package/PKG_ADD:**

It is the file that gives us all the commands that the user will use to run the package.

✓ **Package/PKG_DEL:**

It is the file that has the commands to run when the package is being removed from the users path.

✓ **Package/pre_install.m:**

This is the file that run before installing the package to the users path. This function is called with a single argument and the path where the user wants to install the package.

Package/post_install.m:

This is the file that runs after the installation of the package. This function is called with a single argument and the path where the user wants to install the package.

Package/on_uninstall.m:

This is the file that run before the un-installation of the package of the package. The function is called by a single argument and the path where the user wants to install the package.

Package/inst:

This is the file in the package that contains all the predefined function in the package. They mostly include .m files.

Package/src:

This is the file that contains the code that runs before the installation of the package. The octave package manager executes the `./configure` directory in the package and will call the make file if it exists.

✓ **Package/doc:**

This is the file that contains the documentation of the package. The files will be automatically installed in the users directory.

✓ **Package/bin:**

This is the directory that will be added to the Octave path when the package is loaded in the users path.

2.4 Adding and Removing Packages:

Various packages are present and can be downloaded into the system by Octave-Forge. The packages which are downloaded will be then present in the system downloads folder. For suppose we want to install the image package that is downloaded from forge. The file name is suppose **image-x.x.x.tar.gz**.

Type the command **pkg install image-x.x.x.tar.gz** in the octave prompt. Then if the package is successfully installed nothing will be displayed in the prompt. If errors are occurred while installing the package, it will display the errors in the prompt. If we want to install several packages at once we should type **pkg install** command in the prompt.

To check the number of packages installed in the Octave type **pkg list**. For suppose if we have installed the above image package it will be displayed in the pkg list. If a star mark is present beside the name of the package then it shows that the package is loaded and ready to use.

Introduction

To remove the package that is installed type **pkg uninstall** command and name of the package we wish to uninstall. If we want to uninstall the imager package which we have just installed type **pkg uninstall image** in the octave prompt.

If the package is successfully removed from octave no error will be shown. Or else octave will display the error in the prompt. Once it is uninstalled we will not find the package in the octave packages list. We can type **pkg list** for confirmation.

Some of the packages in octave will have some dependencies like one package depends on another package. If a package is depending on other package octave package manager will check whether the other package is installed or not. If not it will display error in the prompt. Since some packages are dependent on other the functions in those packages may not work. We can turn off the dependencies in packages installation but it is not recommended.

There are several add-on functions present in octave. Each and every command has different action corresponding to the command.

Some of the available commands that have different functionalities are:

install

install command in octave installs the named packages in octave. For example **pkg install image-x.x.x.tar.gz** installs the package found in the file image-x.x.x.tar.gz.

Nodeps

The package will have the ability to get installed even when it is dependent on another package. The package dependency will be disabled by the package manager. WE should not use this command as the functions which are depending on another package may not work.

Forge

By typing octave forge command the package manager will install the package directly from octave forge repository. We should have an Internet connection to perform the operation. For example if we want to download the symbolic package directly from forge simply type **pkg install -forge symbolic**. Then the package manager will automatically download the package from forge repository. But there is some risk involved as the packages are not verified by the package manager before getting installed.

Verbose

The package manager will print the outputs of all the commands that are being performed by the user in octave command prompt.

Update

The manager will check for any outdated items in the package and it will update the items if any of them are found not updated. It requires a internet connection.

It will update all the packages by executing the command **pkg update**.

Uninstall

The package manager will uninstall the name of the package that is being executed by the user. By simply typing the **pkg uninstall communications** it will uninstall the communications package from the octave.

Load

Load command adds the packages that are wanting to be loaded to the path. Then the user will be able to use the named package.

Pkg load arduino loads the arduino package and all its functions for the user.

Introduction

Unload

Package unload command unloads the current package in the octave directory and the user will not longer use the particular package. For example **pkg unload symbolic** command unloads the symbolic package from the user directory.

List

list command shows us what packages are currently installed in the octave directory. Typing the name of the package after **pkg list** command tells us a brief info about the package like version, date of the package installed.

Describe

The command describe gives us a description of the package like what are the date of installation of the package, version.

For example by typing **pkg describe image control** the package manager describes about the image package and control package. If any one those packages are not installed a error will be displayed in the prompt.

local_list

The command local list sets the file to look for information in the locally installed packages. The packages which are installed in local directory will only be available to the current user.

global_list

The command local list sets the file to look for information in globally installed packages. The packages which were installed are available to all users.

Build

Build command builds a binary form of packages. The binary file is itself a new package and we can normally install that particular package

pkg build buildir image-1.0.0.tar.gz builds a binary package. Builddir is name of the particular directory the temporary package will be installed.

Rebuild

The command rebuild rebuilds the package from the database and from the installed directories. We can use this command when the packages have been corrupted or lost.

2.5 History behind IRC:

IRC is a short form for Internet relay chat. It is a internet dependent chat methodology that was developed so that we can be able to contact the developers that are present across the world so that the combined effort of user and the developer gives us some better results.

Jarkho Wiz Oikarinen first wrote IRC client and server. The first part he implemented was chat part where the users can communicate between themselves. It was first implemented in his own university and once it was released outside of his local university range he immediately installed a server which was named as IRC network. Jarkho developed the IRC as a extension for BBS(Bulletin Board System) which is a computer server running software that allows users to connect to a particular server.

Coming to how IRC works, when you login to your IRC with a client application it logs you to a particular server which you have set as a default program. Then you will be able to join a new public chat depending on the program you have chosen. Any message you have sent is being sent to your client through the IRC server and every other user is connected to the same server ie., every user is connected to the same chat room. As the server copies

Introduction

and shares the messages so fast that the an illusion is created that everyone is connected to all other members and in the same chat room through a common server.

Large number of IRC servers are connected together in very huge amount of networks so that they all share the same channels and every other channel then combibes with the same channel making more number of chat rooms. Every server on the chat room is already connected to every other channel in the chat room and every user in the chat room are logged on to their clients and they interm are connected to one big large server.

Some major IRC networks are DALnet, EFnet, QuakeNet.org, Undernet. Of them DAL net is considered to be the most user friendly of all the major IRC networks where the users have the right to command their own chat experience

CHAPTER-3

FREE SOFTWARE CONTRIBUTION

There are several ways to contribute to a free softwares like bug fixing, modifying a function in a package for better result, creating our own package, maintaining a package etc.,

3.1 Bug fixes:

Bug fixes is one of the major activity in any software development. Most of the company's main task is to fix the bugs that are encountered during the execution of various tasks in the software development. To fix the bugs that have occurred a detail study and analysis of the bugs are necessary for efficient performance.

Manually labeling bug fix activities are usually time consuming and there generally requires a good approach to identify the bugs as they may be very hard to track and if once tracked they may become difficult to get fixed. To help identify bugs a detailed bug flow is necessary. The knowledge about the package or function that we are identifying the bugs are necessary for easy identification of bugs.

In the case where you trying to find your own bug and if we are unable to identify those bugs then we can follow the following steps:

1. Enter the bug in a case tracking system:

During the process of identifying there may be two common error that have happened to most of the users. They are:

- (I) Some keywords or points in the bugs may go missing
- (ii) No knowledge on the function or package you are working on

Introduce your self to a bug tracking system named case tracking system and record all the results till date. In this way we can track what the bugs that are identified, prevents you losing track of your current task or any pending works that you have assigned yourself. Always remember to record what we are doing to fix it, what they were expecting, what was happening instead.

2. Search for the error message on line:

The error messages that have occurred during the bug fixing process are actually a blessing in disguise. Because the error can be goggled and if we are lucky enough we can find a solution to that particular bug.

3. Identify the next line of the code:

If there is no luck in searching the web then go to the next line of the error message. Type print() functions in alternate lines of the code and search up to where the process is running. In this way we can at least identify up to where the process is running successfully and identify the region that is not working correctly.

4. Identify the line of the error code:

Once you have verified the next line of the bug code then come back to the actual line where the bug actually occurs and then very often we will identify that the line that the bug is identified has a bad data passed from the above line of code from the stack.

5. Identify the species of the bug:

A bug may be occurred in many forms and they all belong to some categories. They are:

1. Index out of range:

There may be sometimes a error that is occurred that is index out of bounds. This is an exception that usually occurs when you began an error at 1 instead of 0 or .count or .length returns the same value etc.,

2. Race around condition:

A race around condition happens in some of the cases during the identification of bugs in a system. It arises when two or more threads in a class access shared data but try to change the value at the same time.

3. Configuration or constraints are wrong:

We must also take a close look at the constants that are used when we have written a code or program. It usually takes a long time to try to identify this particular error because the constraints have different usage in one language from another. So we often tend to do wrong on those situations.

4. Unexpected null value:

Value is not initialized is one of the most common error that has occurred during the execution. Make sure we are checking for null references while executing.

5. Bad Input:

Giving proper inputs is also one of the most essential operations that should be performed because we cannot give a character value for a integer value on the input and we cannot perform a arithmetic operation through character value.

6. Assignment value errors:

Giving assigned value instead of comparison values are also one of the most frequent errors that will be occurred. We often use = when we need to give ==.

7. Wrong representations:

Using decimal values instead of decimal values, not having a integer big enough to accommodate can also be a simpler bugs that can be solved during bug fixes.

8. Overflow & Index out of range:

If we try to insert data larger than the space allocated for you then you most probably tend to get the above error.

9. Math Error:

Assigning wrong arithmetic operators and formulas also tend to give us errors that will be observed.

10. Function operator errors:

If we are looking to concatenate a string and a integer it wont work. As we have assigned a string value in one variable and integer value in another variable.

Incase your bug is not in the above category then we need to do more work and follow the remaining steps.

6. Use process of elimination:

If the process is becoming very complex and out of range then comment a part of the cod and run the remaining code. Commenting a section of the code disables the code from working and we can see whether the remaining code is working correctly. If the bug is manifesting itself then begin disabling it one by one

and until it works again. Now bring back the commented parts of the code and check for its functionality.

7. Begin logging the program

Divide the code into sections and try running those section of codes and try to identify the occurring errors. Begin the process one section at a time and wait for the function to work incorrectly.

8. Replace components:

Try replacing RAM'S, servers and service packs. Try installing and un-installing the packs and servers and if the bug is rectified then it is either the hardware or the package. If your program has network input and output connections then check for switches, cables and try reinstalling them. Try plugging the power to different supplies and hardware components. If the bug still persists then the problem is definitely in the software.

9. Looking at the dependencies:

Check whether the software you are using is dependent on any other software, then try installing or reinstalling the software and cross verify it.

10. Ask for help:

The last step is to ask others for help as they may know better than us and have more bug fixing capabilities than you. By then we will also have a clear idea on how the bug is happening and it becomes easy for both of them to rectify the bug. If not try posting the problem as a query in the internet and wait for solutions. If any person comes up with a solution then we can use them for our usage.

There are certain things that will definitely help us to solve bugs that is having a second person also to look into our bugs and taking necessary time to solve our bugs and have someone in your group who has good knowledge in programming and who is a efficient debugger.

Free Software Contribution

Make sure you do not get frustrated while solving some bugs as some bugs are very hard to track and we just need to be careful while trying to find the solution. Also take some time to regroup and get some new ideas to solve. Never blame anyone while doing and take one step at a time while doing and never panic in any situation. It is most important to gather your team and sit back and discuss the work done and what needs to be done and try to solve the problem collectively.

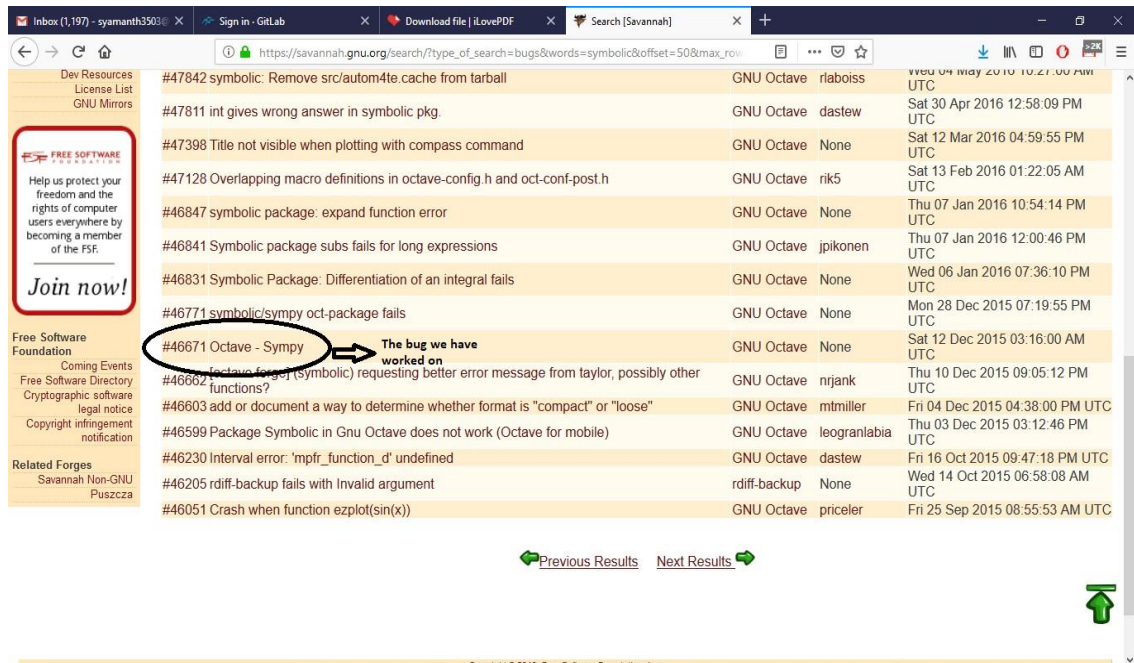
These are the above methods that help us to identify bugs in our own command. But there are bugs that are present internally inside a software or a repository. We can solve those bugs that are already reported by the developers and we can identify them for our own purposes.

The Octave software has several bugs that are already reported by various users across the world. These bugs are classified according to the package. It is also a form of contribution to the open source community as the developers who have identified errors are having problems with those bugs and we can help them in our own way.

There is a repository in octave-forge which shows us the bugs that are available in the repository. Each and every package has number of bugs that are waiting to be solved and select the name of the package we want to identify the bug and then click on search.

Various number of bugs are displayed on screen and we can select a bug of our own choosing and we can start to solve those bugs.

Free Software Contribution



Dev Resources	#47842 symbolic: Remove src/autom4te.cache from tarball	GNU Octave	rlaboiss	Fri 04 May 2016 10:21:00 AM UTC
License List	#47811 int gives wrong answer in symbolic pkg.	GNU Octave	dastew	Sat 30 Apr 2016 12:58:09 PM UTC
GNU Mirrors	#47398 Title not visible when plotting with compass command	GNU Octave	None	Sat 12 Mar 2016 04:59:55 PM UTC
	#47128 Overlapping macro definitions in octave-config.h and oct-conf-post.h	GNU Octave	rik5	Sat 13 Feb 2016 01:22:05 AM UTC
	#46847 symbolic package: expand function error	GNU Octave	None	Thu 07 Jan 2016 10:54:14 PM UTC
	#46841 Symbolic package subs fails for long expressions	GNU Octave	jpikonen	Thu 07 Jan 2016 12:00:46 PM UTC
	#46831 Symbolic Package: Differentiation of an integral fails	GNU Octave	None	Wed 06 Jan 2016 07:36:10 PM UTC
	#46771 symbolic/sympy oct-package fails	GNU Octave	None	Mon 28 Dec 2015 07:19:55 PM UTC
	#46671 Octave - Sympy	GNU Octave	None	Sat 12 Dec 2015 03:16:00 AM UTC
	#46602 octave-forge (symbolic) requesting better error message from taylor, possibly other functions?	GNU Octave	nrjank	Thu 10 Dec 2015 09:05:12 PM UTC
	#46603 add or document a way to determine whether format is "compact" or "loose"	GNU Octave	mtmiller	Fri 04 Dec 2015 04:38:00 PM UTC
	#46599 Package Symbolic in Gnu Octave does not work (Octave for mobile)	GNU Octave	leogranlabia	Thu 03 Dec 2015 03:12:46 PM UTC
	#46230 Interval error: 'mpfr_function_d' undefined	GNU Octave	dastew	Fri 16 Oct 2015 09:47:18 PM UTC
	#46205 rdiff-backup fails with Invalid argument	rdiff-backup	None	Wed 14 Oct 2015 06:58:08 AM UTC
	#46051 Crash when function ezplot(sin(x))	GNU Octave	priceler	Fri 25 Sep 2015 08:55:53 AM UTC

Fig 3.1 Reporting the bugs in symbolic package

Given above are the bugs that are identified in symbolic package. We have then selected the bug which we want to solve

The bug shows **Octave-Sympy**

First we will know what is meant by sympy. Sympy is a python library for symbolic mathematics. Sympy is a short form for symbolic python library. It is a computational library where we perform mathematical operations. Sympy is installed in python with a few dependencies. Sympy features include discrete mathematics, algebra, calculus etc.,

First of all we need to install Octave-4.2.2 version. It is the latest version of the software and it has all the updated features. The octave version has no inbuilt packages and we need to install the packages which we want use.

Since the bug that is identified is related to symbolic package we need to install the symbolic package in Octave by typing the following command in Octave prompt

```
>>pkg install -forge symbolic
```

while installing the package the octave package manager reports the following error like sympy version ≥ 1.0 is required. We can also install the package in terminal using the command **sudo apt-get install octave-symbolic**. Python version above 2.6 is also needed for the package or else you can install python 2.7.0(min) from the browser to work.

The sympy version that get installed automatically is 0.7.4.1. But we need sympy version ≥ 1.0 for the package to function. So go to your browser and download sympy version 1.2 which is available for free. The latest version of sympy is required because we are using the latest version of Octave and it is updated upto date and we don't want to use the outdated version of functions inside the symbolic package.

In order to install the sympy latest version using terminal type the commands below and verify:

```
>> sudo python setup.py install
```

The command checks whether latest version of python is installed or not. If not it will display an error in the window.

```
>>python
```

This command checks whether python is successfully loaded in the directory or not. The python should be loaded in the directory for the package to be installed successfully.

```
>>import sympy
```

The command loads the latest version of sympy that is downloaded and it should load the sympy into the package directory. If not successfully loaded or an error

is occurred an error message will be displayed in the window showing version is not compatible or not loaded successfully.

If error still persists then probably there is still version incompatibility or sympy is not downloaded correctly. Check for sympy in web again and try reinstalling the package.

Symbolic is one of the package that has dependency with the sympy package as various functions in python are performed using the sympy package.

3.2 Package contribution:

There are many sources to contribute to the open source software tools one such source is Package contribution. Creating a package includes developing functions modifying actual package into different package which are created to meet our own requirements in performing an operation. These files or functions are made into a folder which can be installed as our own package. There are many ways to create a package like creating our own package or modifying the existing package available in the internet. To modify a package that is present in the Github we need to search for the required package and fork the package from the authors account and make our own changes and contributions.

Process of forking a package for octave:

- Select a package related to octave.
- Go to the author's repository of the selected package and will be navigated to the git profile of the author.

Free Software Contribution

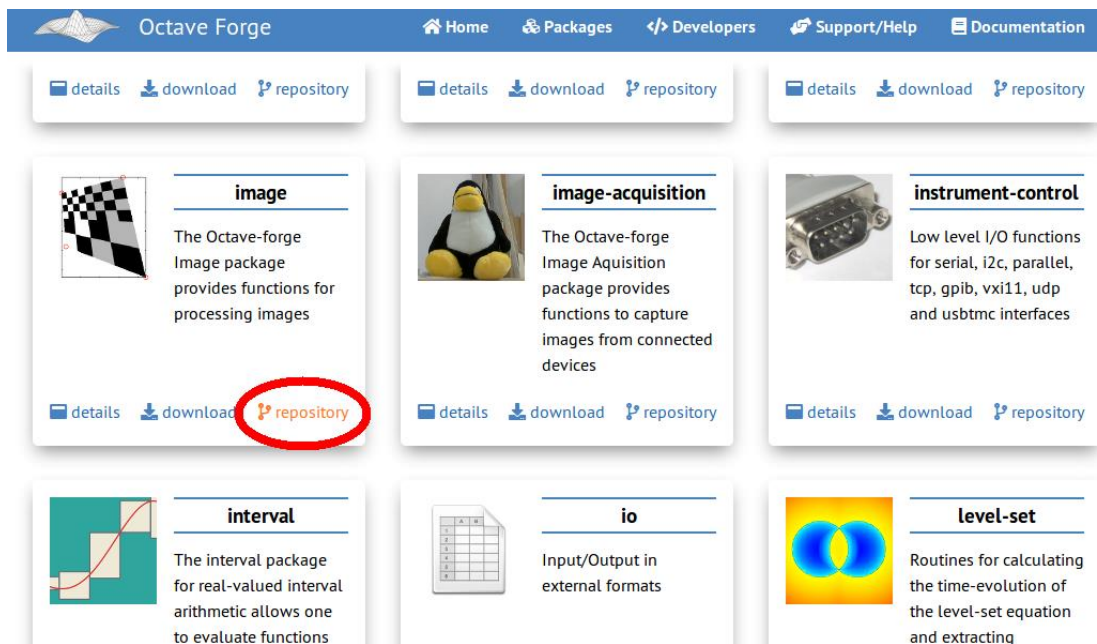


Fig 3.2.1 Showing the repository in octave forge

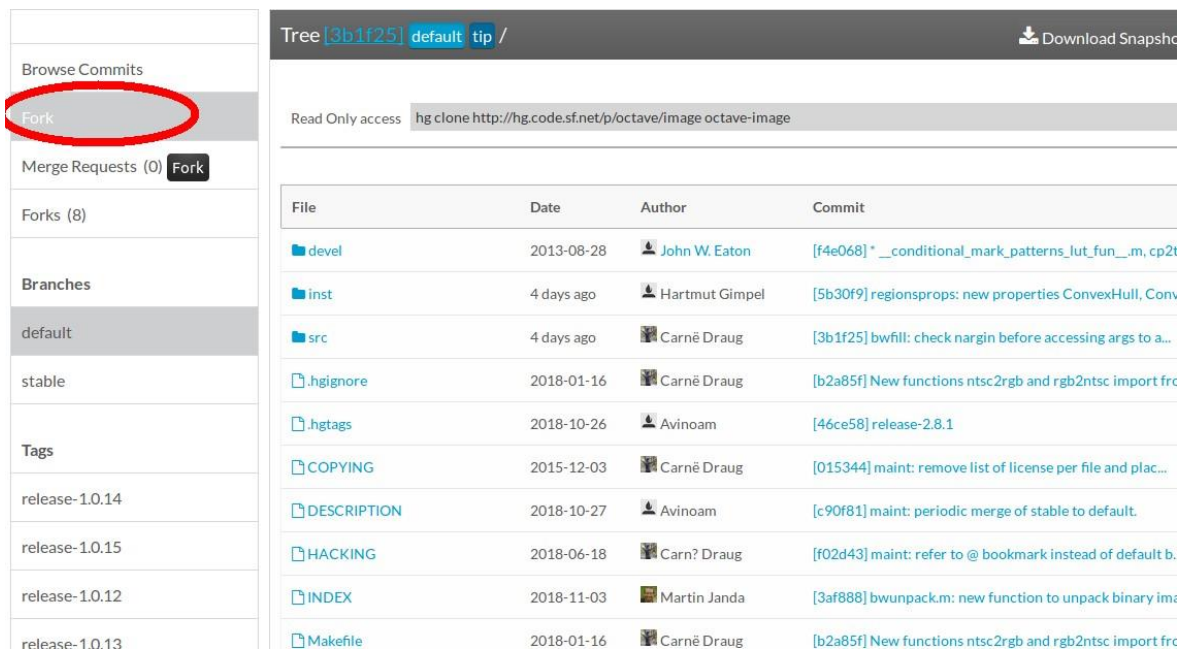


Fig 3.2.2 Showing the repository and forking method

- Then login to the source forge account if you don't have source forge account, create an account and register.

Free Software Contribution

- After getting into the account clone the repository for the contents in the package.

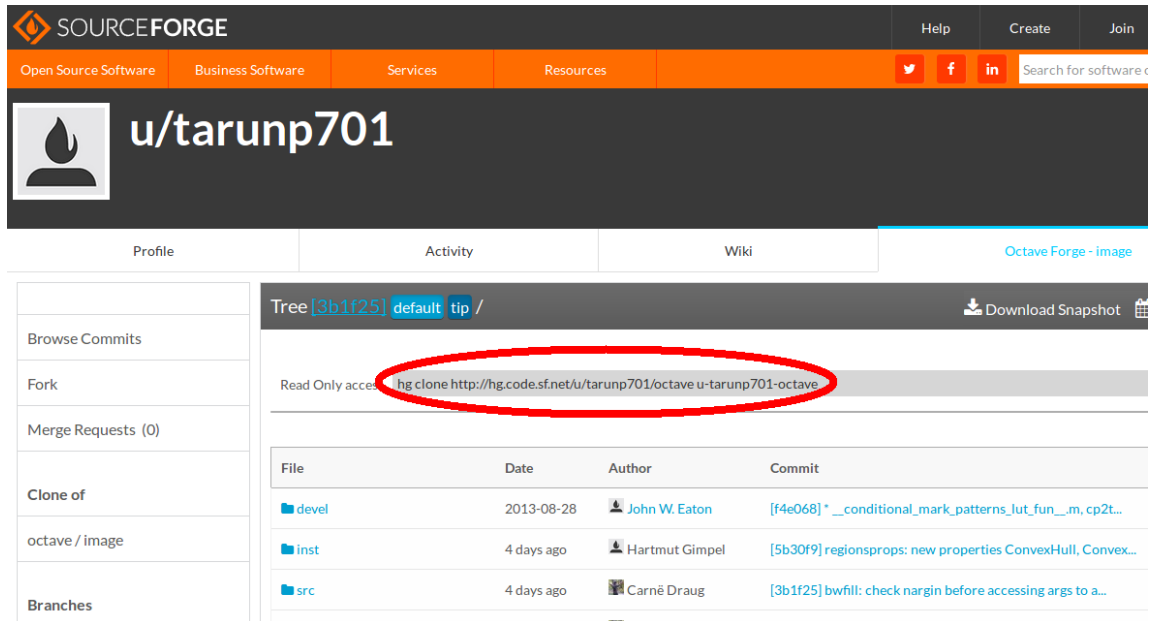


Fig 3.2.3 Cloning the package

- After cloning we are ready to make changes in the package.

Here we are making changes in various functions in a package of image processing. The functions are shown below:

1. Imadjust:

The imadjust function in octave intensifies the values of the image in grayscale from one level to another level. It basically intensifies the contrast of output images.

The command name is **imadjust(I,[low_val high_val])**

The values of low_val and high_val range from 0 to 1. As the output image is contrasted drastically the function is written as imadjust as it adjusted the input image to out required value level.

Sample commands are:

Free Software Contribution

```
Img=imread("image.jpg");
```

```
Imshow(Img);
```

```
Imadjust(Img);
```

By using the `imadjust` function a image is contrasted to a fairly brighter image as shown below. This is actually a contribution to octave as we have forked the image from github directory and we have provided better results than the previous version of the code.

The enhancement of the image is done successfully using the `imadjust` function and we have identified better results than the previous image

The input image we have given is

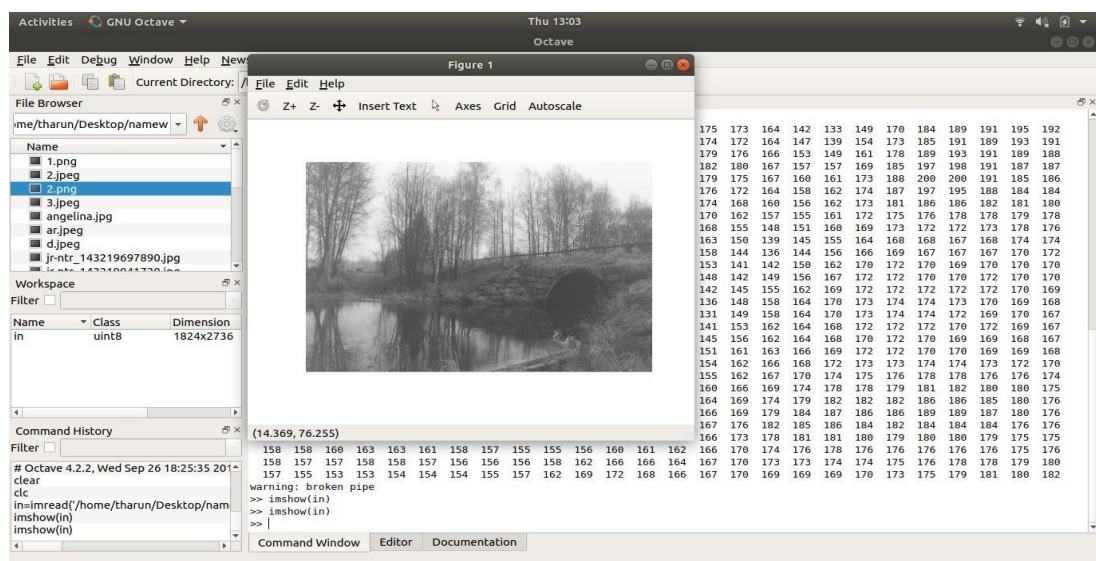
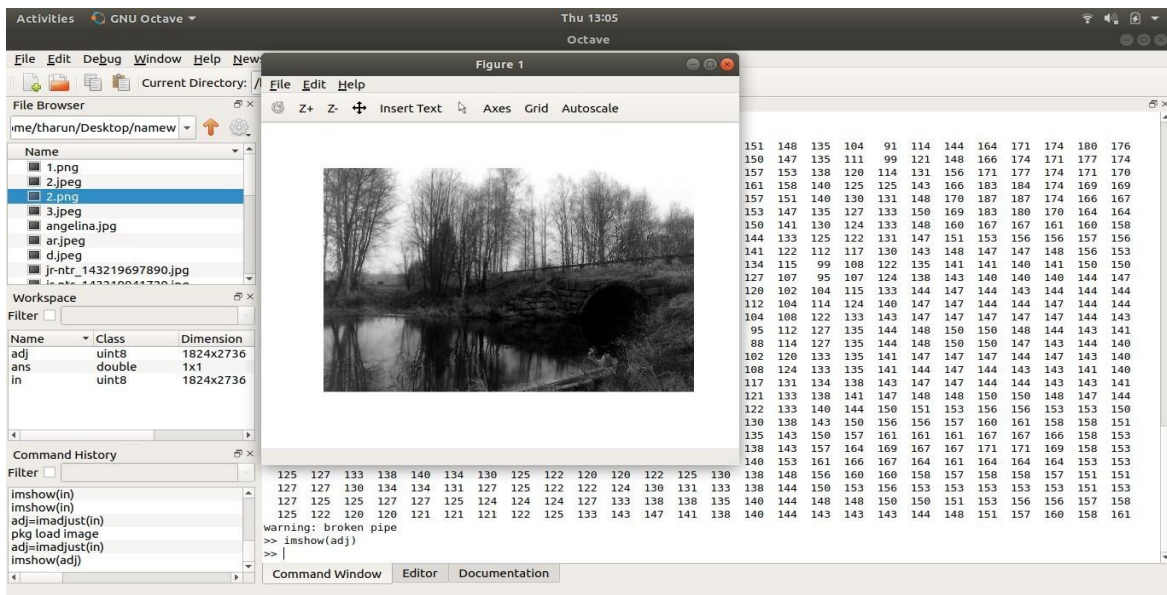


Fig 3.2.4 Input 1 of `imadjust()`

The output image obtained after contrast enhancement:

Free Software Contribution



3.2.5 Output 1 of imadjust()

As we can see there is an increase in contrast of the one image to another image. The input image given is more foggy and unclear and the output image obtained is more clear and visible clearly to the eye. The pixel values have also varied successfully and we have successfully obtained better results by varying the matrix values in the pixels.

Free Software Contribution

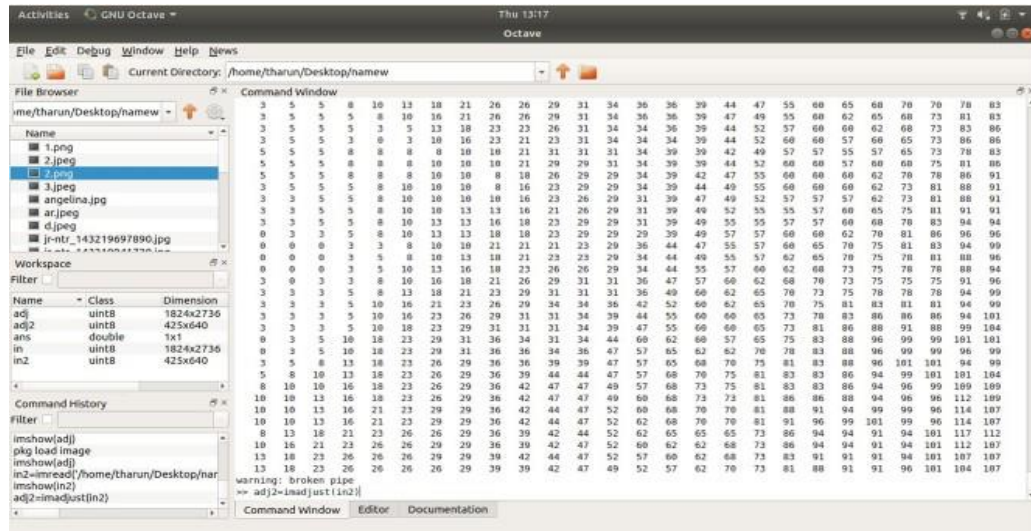


Fig 3.2.6 Output 1 pixel values of imsdjust()

The above image are the pixel values of the output images. The values have increased substantially and we can definitely observe the contrast enhancement of the image.

Another example for contrast enhancement of an image is pictured below. We have taken another picture with low contrast and our main aim is to enhance the image.

The obtained results are

Input image given to the octave:



Fig 3.2.7 Input 2 of imadjust()

Free Software Contribution

The background of the input image is dark and the edges are also very unclear. We can definitely enhance the image using the `imadjust` function.

The output result that is obtained is shown below

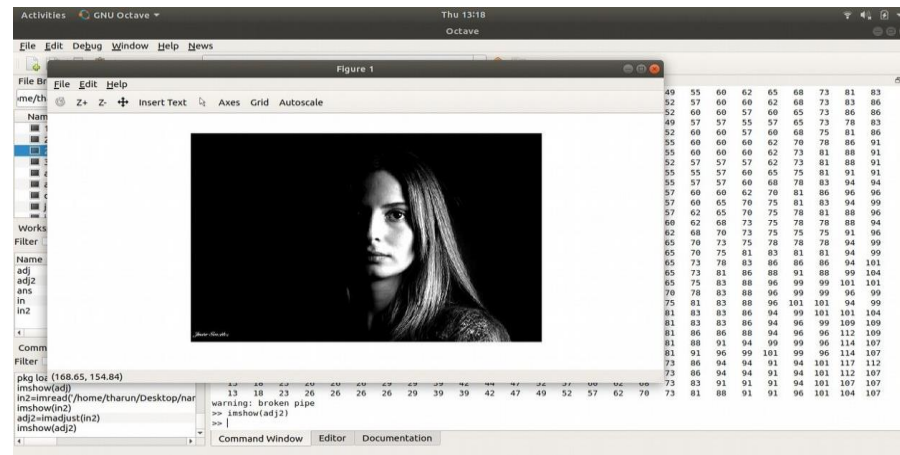


Fig 3.2.8 Output 2 of `imadjust`()

Another function we have taken is `imnoise`().

The `imnoise` function in octave adds noise to an image. Noise is basically an external disturbances in an image. The image which is subjected to noise undergoes some changes compared to the original image. By rearranging the values inside the `imnoise`() function in octave we can see the changes in the original image. The image is subjected to various change in its parameters and we need to get the required image with some extra noise added to it.

Let us test with one sample image. The input image given is shown below:

Free Software Contribution



Fig 3.2.9 Input 1 of imnoise()

This is the original image. This image is now subjected to some changes in the values in the code. The expected output image should have contained more noise compared to the previous image. The image is also said to be less clear and should have lesser contrast compared to the original image.

The gaussian noise is considered for the process and the function for gaussian noise is **Imnoise(img,"gaussian");**

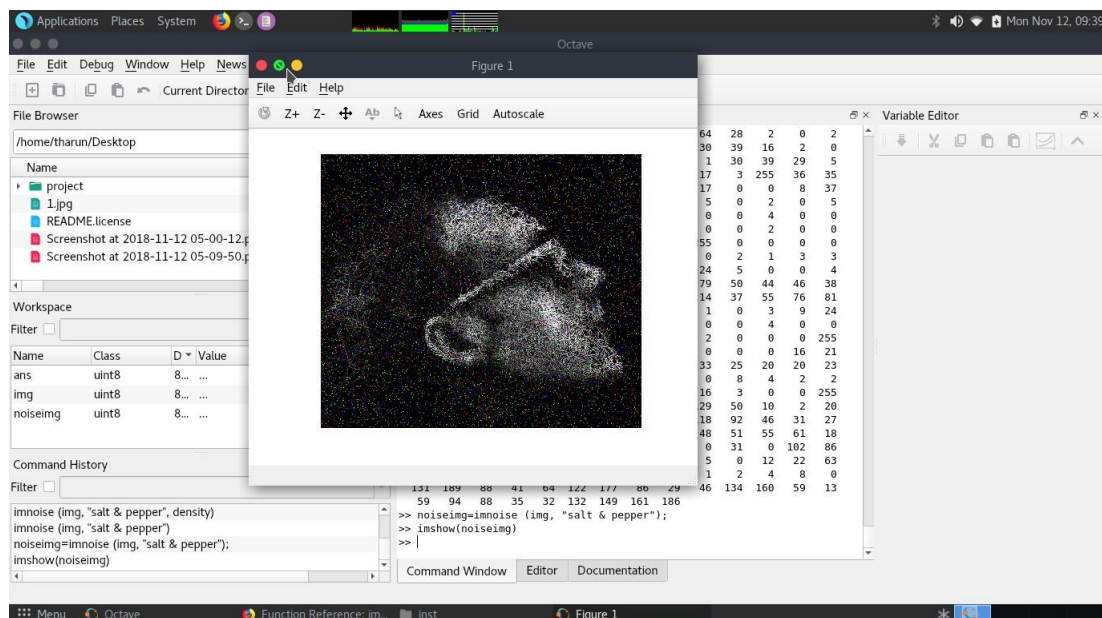


Fig 3.2.10 Output 1 of imnoise()

Free Software Contribution

The gaussian noise is applied to the input image without changing the values inside the main function. We can clearly see the external noise applied to the image and now we need to change its internal values in the function for better noise in the image. The image should be less clearer and have lesser contrast compared to the above obtained image.

Another function we have worked on is ROI in octave. The roi function in octave selects a region of interest in the image based on the colour of the image.

We will read a particular image into the octave and the command then selects the region of interest of image within the low and high regions.

The function is **roicolour(img,low,high)**

The image we have given as a input will select the region between the given low and high values

The input image is given below

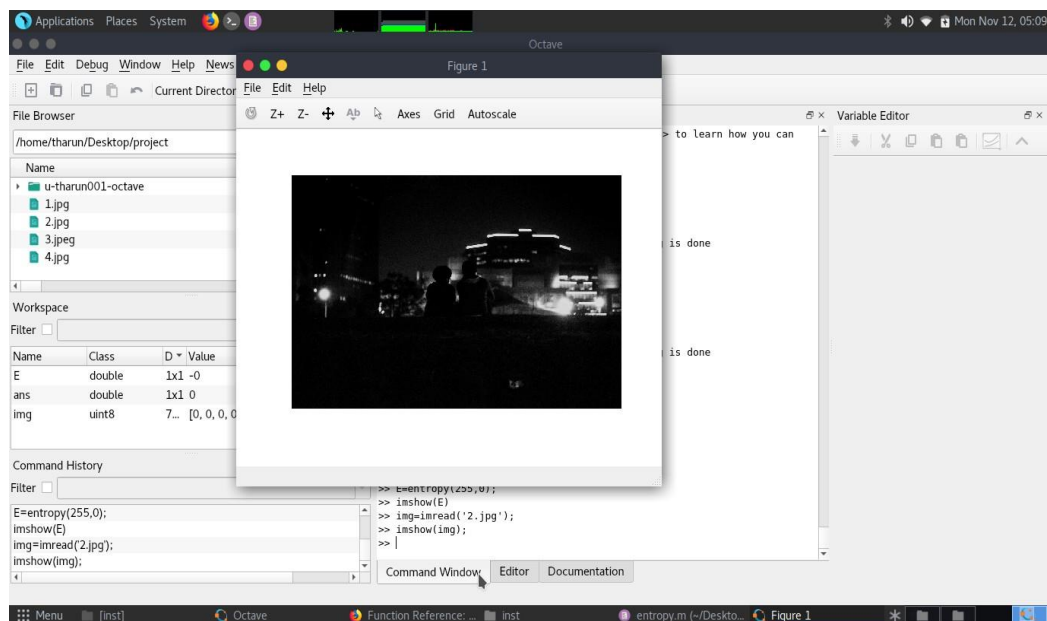


Fig 3.2.11 Input 1 of ROI

The input image is being loaded into the octave through the roi command and the values inside of the function are changed to our knowledge so that the region of interest we want changes to our requirement.

Free Software Contribution

The expected output image should be of low value in the image compared to the input image and the image should be verified.

The output obtained by varying the values are:

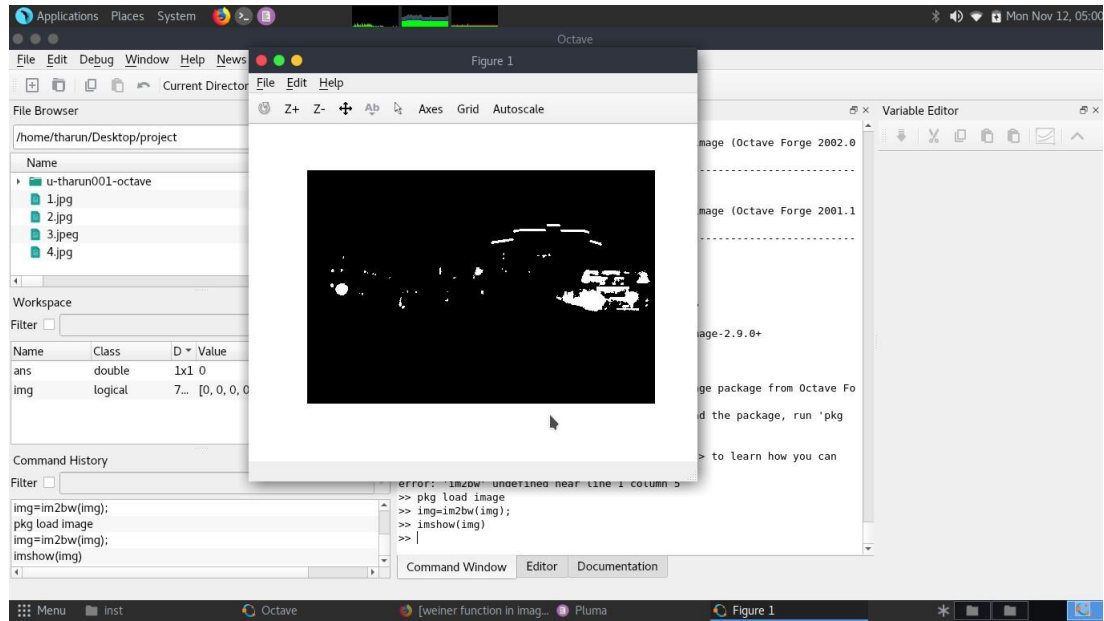


Fig 3.2.12 Output 1 of ROI

As we can see the output image clearly reflects the region of interest and it differs from the input image. The region of interest is very high compared to the input image and the change in values inside the function clearly reflects the output image.

Another function we have worked on is bwmorph

- This is a morphological function which is used to perform several operations on a black and white or binary images. Some of the morphological operations are bothat, bridge, skel, remove etc. Each operation has its own style on morphing an image. The following are some of the valuable methods performed.

Bothat:

- This is one of the morphing method where an image undergoes bottom-hat filtering which means the background parts of the actual image is subtracted from the original image using various structured elements.

Skel:

- This is another type of operation that uses skeletonisation of the image. This runs using the skel-pratt algorithm where the skeletonisation is performed on the nearby edges of the image.

Remove:

- If all the 4- connected pixel neighbors are 1's then the pixel values are set to 0. This is also known as the interior pixel operation.

The original image is shown below:



Fig 3.2.13 Input 1 of bwmorph()

The bottom hat image obtained by changing the parameters inside the original function of the input image is shown below:

Free Software Contribution

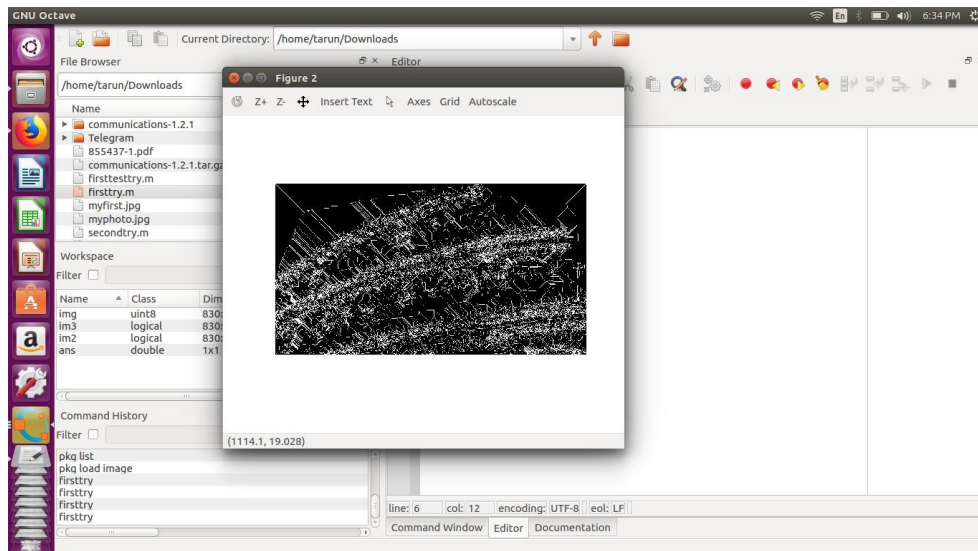


Fig 3.2.14 Output 1 of bwmorph()

The output image obtained after using skel function to the input image is



Fig 3.2.15 Output 2 of bwmorph()

CHAPTER-4

Conclusion

By developing the packages that are used in the software the package updates automatically and the packages will be very easy to use and the software also works efficiently. It is essential to develop the packages and the people who are developing are the developers who develop them and the bugs that are present of them should also be rectified for the better performance of the tools we are using

CHAPTER-5

Future Scope

Creating new packages and fixing the bugs is an endless process and in the future there will be many new packages that will be written and it is essential for the development of the package. Any open source software should undergo changes for the own benefit of the software and the community.

CHAPTER-6

References

- <https://octave.org/doc/v4.2.0/Installing-and-Removing-Packages.html>
- <https://www.gnu.org/software/octave/>
- https://www.google.com/search?source=hp&ei=bY3aW7SdM5H_rQGF0or4Dw&q=what+is+open+source+contribution&oq=what+is+open+source+&gs_l=psy-ab.3.8.0l10.2577.10012.0.13515.27.21.1.4.4.0.169.2262.0j16.16.0...0...1c.1.64.psy-ab..6.21.2297.0..35i39k1j0i67k1j0i131k1j0i131i10k1j0i10k1j0i131i67k1j0i131i10i67k1.0.jAxC5fBfxXs
- <https://opensource.guide/how-to-contribute/>