Detect the Hand Signals by Using Sign Language Recognition

M.umapathy[1] M.Preethi[2] C.Hashmithaa[3] A.Jaisri[4] M.Savitha[5]
[1]Assistant Professor
[1,2,3,4,5]Department of Computer Science and Engineering
[1,2,3,4,5]The Kavery Engineering College, Salem, India

Abstract— Hand gesture is one of the methods used in sign language for non-verbal communication. It is most commonly used by deaf & dumb people who have hearing or speech problems to communicate among themselves or with normal people. Various sign language systems have been developed by many makers around the world but they are neither flexible nor cost-effective for the end users. Hence we introduce software which presents a system prototype that is able to automatically recognize sign language to help deaf and dumb people to communicate more effectively with each other or normal people. Pattern recognition and Gesture recognition are the developing fields of research. Being a significant part in nonverbal communication hand gestures are playing key role in our daily life. Hand Gesture recognition system provides us an innovative, natural, user friendly way of communication with the computer which is more familiar to the human beings. By considering in mind the similarities of human hand shape with four fingers and one thumb, the software aims to present a real time system for recognition of hand gesture on basis of detection of some shape based features like orientation, Centre of mass centroid, fingers status, thumb in positions of raised or folded fingers of hand.

Keywords: Sign Language Nonverbal Communication, Hand Gesture Method, Pattern Recognition, Gesture Recognition

I. INTRODUCTION

Communication is very crucial to human beings, as it enables us to express ourselves. We communicate through speech, gestures, body language, reading, and writing or through visual aids, speech being one of the most commonly used among them. However, unfortunately, for the speaking and hearing impaired minority, there is a communication gap. Sign language is the mode of communication which uses visual ways like expressions, hand gestures and body movements to convey meaning. Sign language is extremely helpful for people who face difficulty with hearing and speaking. Sign languages recognition refers to the conversion of these gestures into words or alphabets of existing formally spoken languages. For past years Visual aids, or an interpreter, are used for communicating with them. However, these methods are rather cumbersome and expensive, and can't be used in an Emergency. This involves simultaneously combining hand shapes, orientations and movement of the hands, arms or body to express the speaker's thoughts. Sign Language consists of finger spelling, which spells out words character by character, and word level association which involves hand gestures that convey the word meaning. Finger spelling is a vital tool in sign language, as it enables the communication of names, addresses and other words that do not carry a meaning in word level association. In spite of this, finger spelling is not widely used as it is challenging to understand and difficult to use. Moreover, there is no universal sign language and very few people know it, which makes it an inadequate alternative for communication. A system for sign language recognition that classifies finger spelling can solve this problem. Various machine learning algorithms are used and their accuracies are recorded and compared in this report.

Thus the conversion of sign language into words by an algorithm or a model can help bridge the gap between people with hearing or speaking impairment and the rest of the world. The objective of this project was to see if neural networks are able to classify signed ASL letters using simple images of hands taken with a personal device such as a laptop webcam. This is in alignment with the motivation as this would make a future implementation of a real time ASL-to-oral/written language translator practical in an everyday situation.

II. EXISTING SYSTEM

In previous system, basically, there are two approaches for sign recognition vision based and sensor based gesture recognition.

Sign language posses the challenge that they ae multi-channel, conveying meaning through many modes at once. But still sign language linguistics are still in their early stages it is already apparent that this makes many of the techniques used by speech recognition unsuitable for sign language recognition. However even lack of transaction tools most public services are not translated to sign which makes mre difficult for deaf and dump peoples. There is no commonly used written form of sign language so all written communication is in the local spoken language. Although sign language recognition are drastically different in many respects they suffer from similar issue such as co-articulation between signs means that a sign will be modified by those either side of it. Frame Separation Method. Difficult to obtain a complete outline of moving object. Large quantity of calculation, sensitivity to noise.

III. PROPOSED SYSTEM

We have detected the contour we can start to discuss the actual algorithm for detecting the fingertips and the number of fingers shown.To achieve this we will compute the convex hull as well as the convexity defect regions of the hand contour. Instead of trying to come up with some technical explanation of those terms I will just show you what that means is practice. If we simply compute the convex hull of the contour above we will end up with the following result. As you can see the hull is a polygon spanned by the hand contour. The red circles indicate the edge points of the hull. It converts sign language to voice output. Recognition can be done by matching templates of hand by considering contour curve shapes.

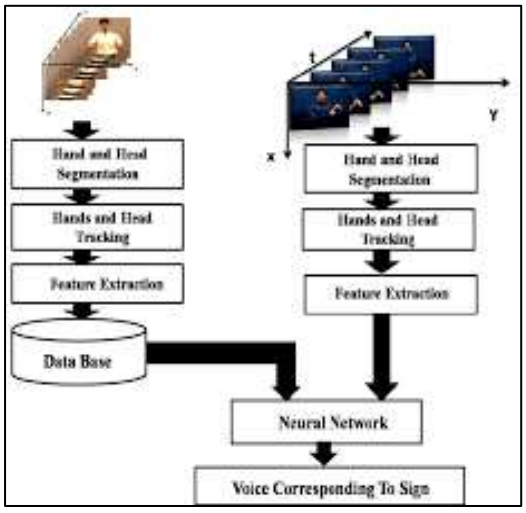

Fig. 3.1: Flow Diagram of Sign Language

The Raspberry Pi gets the first half of its name from a long-standing tradition of using fruit to name new computing systems from classic microcomputers like the Acorn, Apricot and Tangerine to more recognizably modern brands including Apple and BlackBerry but the second half comes courtesy of the Python programming language Open CV introduces a new set of tutorials which will guide you through various functions available in Open CV-Python. This guide is mainly focused on Open CV 3.x version (although most of the tutorials will work with Open CV 2.xalso).A prior knowledge on Python and Numpy is required before starting because they won't be covered in this guide. Especially, a good knowledge on Numpy is must to write optimized codes in Open CV-Python Image Processing − It focuses on image manipulation. Pattern Recognition − It explains various techniques to classify patterns. Photo grammetry − It is concerned with obtaining accurate measurements from images. A class has three parts, name at the top, attributes in the middle and operations or methods at the bottom. In large systems with many related classes, classes are grouped together to create class diagrams. Different relationships between classes are shown by different

Manual testing is the process of testing software by hand to learn more about it, to find what is and isn't working. This usually includes verifying all the features specified in requirements documents, but often also includes the testers trying the software with the perspective of their end users in Manual test plans vary from fully scripted test cases, giving testers detailed steps and expected results, through to high-level guides that steer exploratory testing sessions Automation testing is the process of testing the software using an automation tool to find the defects. In this process, testers execute the test scripts and generate the test results automatically by using automation tools. Some of the famous automation testing tools for functional testing are QTP/UFT and Selenium. Verification is the process, to ensure that whether we are building the product right i.e., to verify the requirements which we have and to verify whether we are developing the product accordingly or not. Activities involved here are Inspections, Reviews, Walk troughs. White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. These White-box testing techniques are the building blocks of white-box testing, whose essence is the careful testing of the application at the source code level to prevent any hidden errors later on. These different techniques exercise every visible path of the source code to minimize errors and create an error-free environment. The whole point of white-box testing is the ability to know which line of the code is being executed and being able to identify what the correct output should be.

Unit testing. White-box testing is done during unit testing to ensure that the code is working as intended, before any integration happens with previously tested code. White-box testing during unit testing catches any defects early on and aids in any defects that happen later on after the code is integrated with the rest of the application and therefore prevents any type of errors later on.

Integration testing ,White-box testing at this level are written to test the interactions of each interface with each other. The Unit level testing made sure that each code was tested and working accordingly in an isolated environment and integration examines the correctness of the behaviour in an open environment through the use of white-box testing for any interactions of interfaces that are known to the programmer.

Regression testing. White-box testing during regression testing is the use of recycled white-box test cases at the unit and integration testing levels.White-box testing's basic procedures involve the understanding of the source code that you are testing at a deep level to be able to test them. The programmer must have a deep understanding of the application to know what kinds of test cases to create so that every visible path is exercised for testing.

Fig. 3.2: Hand Signals

This method of test can be applied to virtually every level of software testing: unit, integration, system and acceptance. It typically comprises most if not all higher level testing, but can also dominate unit testing as well. Functional testing differs from system testing in that functional testing "verifies a program by checking it against ... design document(s) or specification(s)", while system testing "validate a program by checking it against the published user or system requirements" In software engineering, performance testing is in general testing performed to determine how a system performs in terms of responsiveness and stability under a particular workload.

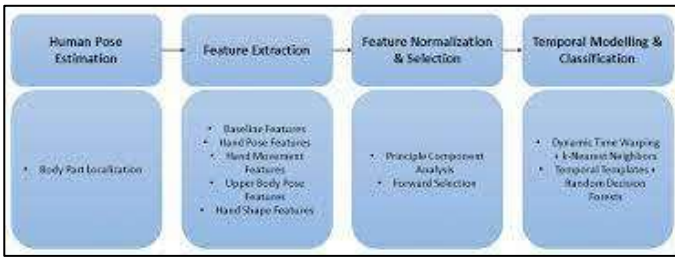

Fig. 3.3: Sign Language Framework

It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage. Performance testing is a subset of performance engineering, an emerging computer science practice which strives to build performance into the implementation, design and architecture of a system. Load testing is the simplest form of performance testing. A load test is usually conducted to understand the behaviour of the system under a specific expected load. This load can be the expected concurrent number of users on the application performing a specific number of transactions within the set duration It essentially involves applying a significant load to a system for an extended, significant period of time. The goal is to discover how the system behaves under sustained use. Spike testing is done by suddenly increasing the number of or load generated by, users by a very large amount and observing the behaviour of the system. The goal is to determine whether performance will suffer, the system will fail, or it will be able to handle dramatic changes in load. Functional testing is a quality assurance (QA) process and a type of black box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (not like in white-box testing). Functional Testing usually describes what the system does. It is also essential to implement a sustainable process for ensuring that test case failures are reviewed daily and addressed immediately if such a process is not implemented and ingrained into the team's workflow, the application will evolve out of sync with the unit test suite, increasing false positives and reducing the effectiveness of the test suite. Unit testing embedded system software presents a unique challenge: Since the software is being developed on a different platform than the one it will eventually run on, you cannot readily run a test program in the actual deployment environment, as is possible with desktop programs It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing. The Big Bang method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing.

A type of Big Bang Integration testing is called Usage Model testing. Usage Model Testing can be used in both software and hardware integration testing. The basis behind this type of integration testing is to run user-like workloads in integrated user-like environments. In doing the testing in this manner, the environment is proofed, while the individual components are proofed indirectly through their use. Usage Model testing takes an optimistic approach to testing, because it expects to have few problems with the individual components. The strategy relies heavily on the component developers to do the isolated unit testing for their product. The goal of the strategy is to avoid redoing the testing done by the developers, and instead flesh-out problems caused by the interaction of the components in the environment The process is repeated until the component at the top of the hierarchy is tested. All the bottom or low-level modules, procedures or functions are integrated and then tested. After the integration testing of lower level integrated modules, the next level of modules will be formed and can be used for integration testing. This approach is helpful only when all or most of the modules of the same development level are ready. This method also helps to determine the levels of software developed and makes it easier to report testing progress in the form of a percentage. The main advantage of the Bottom-Up approach is that bugs are more easily found. With Top-Down, it is easier to find a missing branch link. The assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers. Contrast with verification. The evaluation of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an internal process. Contrast with validation Verification is intended to check that a product, service, or system (or portion thereof, or set thereof) meets a set of initial design specifications. In the development phase, verification procedures involve performing special tests to model or simulate a portion, or the entirety, of a product, service or system, then performing a review or analysis of the modelling results. In the post-development phase, verification procedures involve regularly repeating tests devised specifically to ensure that the product, service, or system continues to meet the initial design requirements, specifications, and regulations as time progresses.

Fig. 3.4: Input with Output

Validation is intended to check that development and verification procedures for a product, service, or system (or portion thereof, or set thereof) result in a product, service, or system (or portion thereof, or set thereof) that meets initial requirements. For a new development flow or verification flow, validation procedures may involve modelling either

flow and using simulations to predict faults or gaps that might lead to invalid or incomplete verification or development of a product, service, or system (or portion thereof, or set thereof). A set of validation requirements, specifications, and regulations may then be used as a basis for qualifying a development flow or verification flow for a product, service, or system (or portion thereof, or set thereof). Additional validation procedures also include those that are designed specifically to ensure that modifications made to an existing qualified development flow or verification flow will have the effect of producing a product, service, or system (or portion thereof, or set thereof) that meets the initial design requirements, specifications, and regulations; these validations help to keep the flow qualified. It is a process of establishing evidence that provides a high degree of assurance that a product, service, or system accomplishes its intended requirements. This often involves acceptance of fitness for purpose with end users and other product stakeholders. This is often an external process.

IV. CONCLUSION

In this project, we have implemented automatic sign language gesture recognition, system in real-time, using tools learnt in computer vision and machine learning. We learned about how sometimes basic approaches. System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic. As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any applicable hardware system(s).

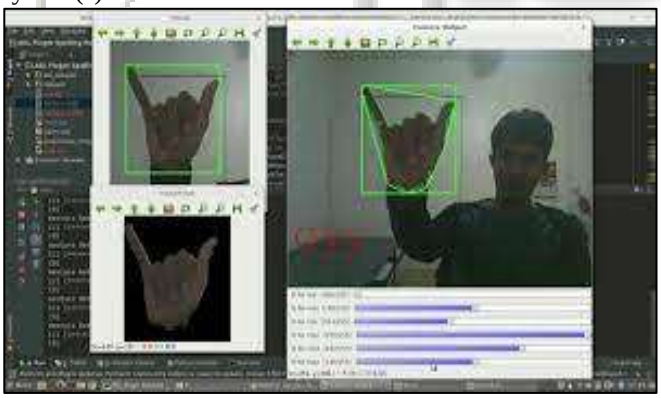

Fig. 4.1: Output

The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole. System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing tests not only the design, but also the behaviour and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification. Despite trying to use a smart segmentation algorithm, the relatively basic skin segmentation model turned out to extract the best skin masks. We also realized the time constraints and difficulties of creating a dataset from scratch. Looking back, it would have been nice to have had a dataset already to work off of. Some letters were harder to classify. To obtain the intended benefits from unit testing, rigorous discipline is needed throughout the software development process. It is essential to keep careful records not only of the tests that have been performed, but also of all changes that have been made to the source code of this or any other unit in the software. Use of a version control system is essential. If a later version of the unit fails a particular test that it had previously passed, the version-control software can provide a list of the source code changes (if any) that have been applied to the unit since that time Software Testing is an art which helps in strengthening the market reputation of a company by delivering the quality product to the client as mentioned in the requirement specification documents. Due to these reasons software testing becomes very significant and integral part of Software Development process.

V. FUTURE ENHANCEMENT

In future work the proposed system can be developed and improved using advanced technologies. We will try to convert all recognize signs to sign language to motion for easier understanding. User dependent model using pre-training the model on a larger dataset (e.g. ILSRVC), that consists of around 14,000 classes, and then fine-tuning it with ISL dataset, so that the model can show good results even when trained with a small dataset. For user- dependent, the user will give a set of images to the model for training, so it becomes

More familiar with the users. According to this way this model will perform well for the particular handled user.

REFERENCE

[1] Ibraheem N.A., Khan R.Z. Vision based gesture recognition using neural networks approaches: a review. International Journal of human Computer Interaction (IJHCI). 3(1), (2012).
[2] SymeonidisK., Hand Gesture Recognition Using Neural Networks, MS Thesis. University of Surrey, UK, (2000).
[3] Ranganath C.W. Ng, S. Real-time gesture recognition system and application, Image and Vision Computing, Vol. 20, Issues 13-14, pp. 993-1007, (2002).
[4] Licsar, Sziranyi T.Dynamic training of hand gesture recognition system, IEEE, 0-7695-2128-2/04, (2004).
[5] Ivekovic S., and Trucco E. Human body pose estimation with PSO. IEEE Congress on Evolutionary Computation Sheraton Vancouver Wall Centre Hotel. Vancouver, BC, Canada, (2006).
[6] Mitra S., Acharya T. Gesture Recognition – A survey, IEEE Transactions on systems, man, and cybernetics—Part C: Application and Reviews, Vol. 37, No. 3, (2007).
[7] Marcel S. ―Hand Posture Recognition in a Body CenteredSpace‖, IEEE, (1999).
[8] Schlenzig J. Hunter E., and Jain R. Recursive spatio-temporal analysis: Understanding Gestures. Technical

report, Visual Computing Laboratory, University of San Diego, California, (1995).
[9] Grant H., Lai C. K., Simulation modeling with artificial reality technology (smart): an integration of virtual reality simulation modeling. Proceedings of the Winter Simulation Conference, (1998).
[10] Starner T., and Pentland A., Real-time American sign Language recognition from video using hidden markov models. Technical Report No. 375, M.I.T Media Laboratory Perceptual computing

