Automated Code Generation System (ACGS)

Apurva Prakash Bhandari[1], Bilva Mehendale[2], Deepali Bhanage Naik[3], Supriya Jagtap[4]

[1,2]Student, Dept. of Information Technology, PES's Modern College of Engineering, Pune, Maharashtra, India
[3,4] Project Guide, Dept. of Information Technology, PES's Modern College of Engineering, Pune, Maharashtra, India

---***---

Abstract - *Knowledge of coding is essential but at the same time acts as a barrier for the one who lacks in it and is time-consuming either ways. Having said that the idea behind this workspace is very much originated from the same baseline and for the same kind of people. We are aware of the process of building any project, Information Gathering: which won't make us wrong if we say it has become automated, logic building and documentation, coding: needs special skills and a specific time-frame, testing: there are a number of applications developed for automated testing and deployment: it is the manual step. The above steps clearly show that the maximum of things of the project development process is automated but when it comes to coding it needs specific skills, time, and engagement of resources (man-power and others) means the cost of the project is increased. Now, all of these problems get the solution if we make it automated with the present project "Automated Coding System". With Applied Algorithms and pattern recognition it is possible to convert any type of logical diagram like a flow chart into code. So, by using machine vision with applied algorithms and correct pattern recognition the process of coding is made automated.*

Key Words: Machine vision, thresholding, contouring, pytesseract.

1.INTRODUCTION

After an indepth analysis of the process of a product in making, the actual time we spend in coding and the same for logic which is drawn on paper in the layout on of the project is a huge. The main logic is already built, it all takes a large amount of knowledge to convert it into a code that works, this limits a few people. Instead if we modify the rules which we already have in place for flowchart[1]. We can convert that itself in code.

Given below are the few rules which we have to have in order for the algorithm to convert the flowchart[2] into a code:

1. When we are declaring the variable in the flow chart, we have to specify the data type by initializing it at the same time.

```
eg:     SET     A=0      //this means int A
        SET     A=0.0    //this means float A
        SET     A= "0"   //this means char A
```

2. When there is supposed to be a print statement, we will imply this using the keyword, "Write" followed by the content in the print statement.

```
Eg:     WRITE "DONE"
```

3. When there is condition where there is supposed to be an input from the use, the use of the keyword "Read" is to be done.

```
Eg:     READ   A
```

4.Conditional statement If remains the same when it comes to notation and keyword i.e. "IS (condition)?". It is written in a diamond shape.

Eg: IS A<0?

4.1. The right side on the diamond will specify the action done when the given condition is true.

4.2. The left side on the diamond will specify the steps to be taken if the condition in the diamond are not met.

4.3. In future the program will merge to a common action.

5. While condition of the flowchart will remain with the same keyword as per the notation followed by the condition, while "condition"

```
Eg:     WHILE A<0
```

5.1. The steps which are to be done while the condition in the "while" loop is true are to be written on the right side.

5.2. The step done after the condition in the "while" loop does not remain valid are written on the left side on the diamond.

6. To end the program, we have used the keyword "Stop".

```
Eg:     STOP
```

7.All the keywords and the alphabets used in the flowcharts have to be capital.

With the above rules been followed correctly, the flow chart will make complete sense to the algorithm, and depending upon the positioning of elements it will convert into a run time code with extend of choosing (in this case).

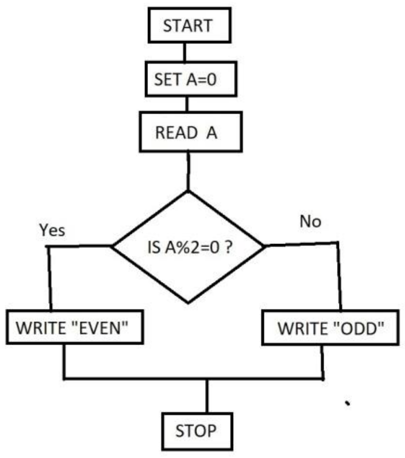


Fig-1: Original Image of the flowchart.

2.AUTOMATION

2.1 Blurring +Thresholding of the Original Flow chart image.

In image processing, a Gaussian blur also known as Gaussian smoothing is the result of blurring an image by a Gaussian function named after mathematician and scientist Carl Friedrich Gauss. It is a widely used effect in graphics software, typically to reduce image noise and reduce detail[3]. The visual effect of this blurring technique is a smooth blur resembling that of viewing the image through a translucent screen, distinctly different from the bokeh effect produced by an out-of-focus lens or the shadow of an object under usual illumination. Gaussian smoothing is also used as a pre-processing stage in computer vision algorithms in order to enhance image structures at different scales.

Thresholding[4],[5] is the basic segmentation method. Basic thresholding can be divided to three methods: binary, truncate and threshold to zero. These methods can be used with inverting function. The advanced techniques of thresholding are the band and multispectral thresholding. Thresholding usually uses grayscale image. In special cases, if we cannot extract the data from a grayscale image, the multispectral threshold is used. Multispectral images are created from color image with three separate channels(R-Red,G-Green,B-Blue). These 2 steps combined will ensure the uniformity of every image while it is processed. As machine vision is known to be so much dependent on the light and texture of the paper. Above filters ensure the reduction of these factors and adjust the values of the blur filter as well the thresholding filter, we calibrate it to any condition of light and texture as we please. The output of these filters ensures the highlighting of the words and the boxes including the diamond in the image so it is easier to detect them for image processing.

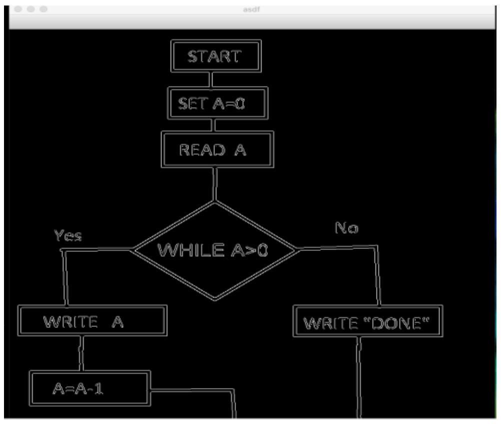


Fig .2. Result after blurring and thresholding

2.2. Contour detection

Contour detection[1] is very strongly tied with optimal thresholding. Contour detection extracts the quantity and shape of objects detected in the image[5]. The next task of contour detection is the description of object hierarchy inside the object (holes, defects, cuts ...etc.). In the case of the original image, as we can see we have to target the boxes and the diamond. While these are being detected there are multiple things going on in the background which is processing the contoured boxes to extract information from them:

1. Boxes / Shapes are being separated from the main images.
2. The content(textual) of the extracted boxes is decoded with pytesseract.
3. The content (positional) of the extracted boxes is decoded and the properties like the placement of the box and the area of the box with respect of the original image are stored.
4. Taking into account the corners of the box, the shape is determined which is used to parse the further algorithm.

3. CODE GENERATION.

3.1 Content Extraction using Tesseract.

After the image is processed, the image needs to be decoded for code generation. For text recognition, it is using the google API called tesseract. Tesseract[6] is an optical character recognition engine for various operating systems. It is a free software, released under the Apache

License.

The boxes extracted from the automation process bare a lot of information to work on and have the main role to change the flowchart into the code. Every property makes the algorithm decide which part of the code the box depicts.

3.2 Content Extraction using Tesseract when the box is square/rectangle.

While the extraction of the box[3] takes place the position of the box w.r.t to the original images is mapped. With it also the property of the box related to the existing corner is taken into account. If the corners are equidistant from each other, aligned, and are sharp then it is detected as a square. If the box is square then according to the rules given in the introduction sector the algorithm knows the content that would be present in it, it is either a read, write, declaration or a process.

The next part identifies what text does the square contain. Tesseract, if installed and used properly with the features enlisted, makes the textual extraction part very easy with printed text and is less ambiguous too. The keyword is separated and the action is taken based on the same. If the keyword points to a declaration or initialization statement, the information of a new variable with its existing value is appended in an array which is further used during the stamens like Read and Write. If the boxes indicate being a process the statement is broken down into pieces to be read correctly depending upon the postfix rules.

Eg: SUM A+B will be broken down into SUM and A+B which then is further broken down into +AB.

3.3 Content Extraction using Tesseract when the box is Diamond.

If the extracted box has Sharp but not in line corners then the box is a diamond which means it is a condition statement, either an IF statement or a While statement. This is taken care by referring to the keywords which are IS followed with a question mark '?' and WHILE respectively. The direction of the representation of the condition being true and not are given in the Introduction chapter already.

What counts more here is the traversing of the contouring process. Typically, the contours follow the natural order from top to bottom which is straight forward as the structure of the flowchart is from top to bottom. But when it includes a condition the program splits in 2 directions one which flows to the condition being true and the other to the false and then the code merges into a singular function again. Here the contours will change the direction from left to right.

1. The algorithm notes the position of the diamond then moves to the left which are the steps that take place if the given condition is true.

2. The algorithm then takes into account of the left most contour converts it into its corresponding code.

3.Moves to the right most contour, stores the information of this contour in a buffer which will be converted in the respective code after the left side of the flowchart is converted i.e. if the steps which are followed when the condition is true take place.

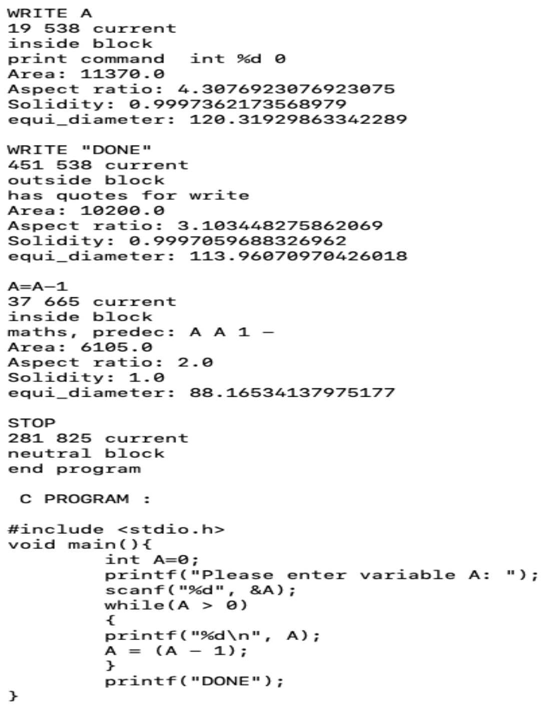
4. After left and right sides of the flowchart are traversed, the contour moves to the center where the merging happens, every time a contour is detected the position is compared to the position of the diamond box if they are below each other then it is declared to be the merging statement.

5. Algorithm neglects the merging statement and converts all the information from the buffer into code.

6. When the buffer is empty the merging statement is converted too.

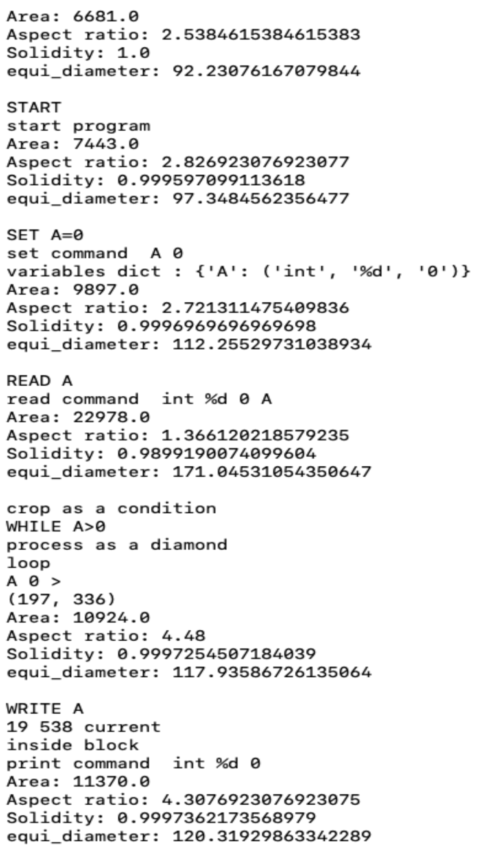
4.OUTPUT

Fig.1. is considered as the input to which the step-by-step processing and working on the algorithm is given in the Snapshots below, it also contains the output code generated by the algorithm. The snapshot in detail explains all the steps and properties which were extracted from the image flowchart.



```
WRITE A
19 538 current
inside block
print command  int %d 0
Area: 11370.0
Aspect ratio: 4.3076923076923075
Solidity: 0.9997362173568979
equi_diameter: 120.31929863342289

WRITE "DONE"
451 538 current
outside block
has quotes for write
Area: 10200.0
Aspect ratio: 3.103448275862069
Solidity: 0.9997059688326962
equi_diameter: 113.96070970426018

A=A-1
37 665 current
inside block
maths, predec: A A 1 -
Area: 6105.0
Aspect ratio: 2.0
Solidity: 1.0
equi_diameter: 88.16534137975177

STOP
281 825 current
neutral block
end program

 C PROGRAM :

#include <stdio.h>
void main(){
        int A=0;
        printf("Please enter variable A: ");
        scanf("%d", &A);
        while(A > 0)
        {
        printf("%d\n", A);
        A = (A - 1);
        }
        printf("DONE");
}
```

Fig.3. Algorithm Process Output with the Final Output



```
Area: 6681.0
Aspect ratio: 2.5384615384615383
Solidity: 1.0
equi_diameter: 92.23076167079844

START
start program
Area: 7443.0
Aspect ratio: 2.826923076923077
Solidity: 0.999597099113618
equi_diameter: 97.3484562356477

SET A=0
set command  A 0
variables dict : {'A': ('int', '%d', '0')}
Area: 9897.0
Aspect ratio: 2.721311475409836
Solidity: 0.9996969696969698
equi_diameter: 112.25529731038934

READ A
read command  int %d 0 A
Area: 22978.0
Aspect ratio: 1.366120218579235
Solidity: 0.9899190074099604
equi_diameter: 171.04531054350647

crop as a condition
WHILE A>0
process as a diamond
loop
A 0 >
(197, 336)
Area: 10924.0
Aspect ratio: 4.48
Solidity: 0.9997254507184039
equi_diameter: 117.93586726135064

WRITE A
19 538 current
inside block
print command  int %d 0
Area: 11370.0
Aspect ratio: 4.3076923076923075
Solidity: 0.9997362173568979
equi_diameter: 120.31929863342289
```

Fig.4. Algorithm Process Output with the Final Output

5. CONCLUSION

Automated Code Generation System is a combination of machine vision and applied algorithm with pattern recognition that convert a simple .jpeg-based image into a code step-by-step. When programming is an automation of a process then this in its true sense is automation of programming. Implying more usages by combining more effective machine learning techniques into the algorithm will give us a fine-tuned tool which can in future with added rules and regulation, be cable of converting any type of GUI as well as UML diagram into its corresponding code.

The introduced machine vision system and applied algorithm for now can be used to convert any flowchart into a code and can be used in school and colleges for checking students works as well as in module wise product development.

ACKNOWLEDGEMENT

It gives me immense pleasure in presenting a paper on ACGS. Firstly, I would like to express our gratitude to our guide and Head of Department of Information Technology, Prof. Mrs. S.D. Deshpande, Prof. Mrs. Deepali Bhanage Naik and Prof.Miss. Supriya Jagtap for their co-operation and encouragement which helped during the completion of this paper as well the project. Their constant guidance and advice played a very important role. They were always available to provide any help and support that was needed, which was crucial in making this paper as detailed as possible. I would also like to thank Bilva Mehendale as she was my paper in the project and her help regarding testing and her suggestions have proved to be very important and conclusive for the project and the paper. Also, I wish to thank my Principal, Prof. Dr. Mrs. K. R. Joshi and all faculty members for their whole hearted cooperation for completion of this paper. I also wish to thank our laboratory assistants for their valuable help in laboratory

REFERENCES

1. N. C. Soni, D. A. Pawar, N. S. Tambe and R. V. Deolekar, "Automated system for code generation from unstructured algorithm," *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, 2016, pp. 1065-1070.M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.
2. Wu, Xiang-Hu & Qu, Ming-Cheng & Liu, Zhi-Qiang & Li, Jian-Zhong. (2011). Research and Application of Code Automatic Generation Algorithm Based on Structured Flowchart.JSEA. 4. 534-545. 10.4236/jsea.2011.49062.
3. Gaussian blur. https://en.wikipedia.org/wiki/Gaussian_blur. May 2020.
4. Thresholding (Image processing). https://en.wikipedia.org/wiki/Thresholding_(image_processing). May 2020.
5. Zidek, Kamil & Hošovský, Alexander. (2014). Image thresholding and contour detection with dynamic background selection for inspection tasks in machine vision. International Journal of Circuits, Systems and Signal Processing. 8. 545-554.
6. Tesseract(Software). https://en.wikipedia.org/wiki/Tesseract_(software). May 2020

BIOGRAPHIES

Apurva Prakash Bhandari,
Student,
Dept. of Information Technology,
P.E.S's Modern College of Engineering,
Pune, Maharashtra, India

Bilva Mehendele,
Student,
Dept. of Information Technology,
P.E.S's Modern College of Engineering,
Pune, Maharashtra, India