International Journal of Computer Science
and Engineering (IJCSE)
ISSN(P): 2278-9960; ISSN(E): 2278-9979
Vol. 5, Issue 2, Feb - Mar 2016, 63-72
© IASET

International Academy of Science, Engineering and Technology
IASET Connecting Researchers; Nurturing Innovations

DEVELOPMENT OF AN AUGMENTED REALITY-BASED G-CODE GENERATOR IN A VIRTUAL CNC MILLING SIMULATION

HWA JEN YAP[1], YUN SUEN PAI[2], SIOW-WEE CHANG[3] & KEEM SIAH YAP[4]

[1 2]Department of Mechanical Engineering, Faculty of Engineering, University of Malaya, Kuala Lumpur, Malaysia

[3]Institute of Biological Sciences, Faculty of Science, University of Malaya, Kuala Lumpur, Malaysia

[4]Department of Electronics and Communication Engineering, University Tenaga National, Selangor, Malaysia

ABSTRACT

In a computer numerical control (CNC) milling process, simulation is integral to ensure that the resulting work piece has very little error. This boils down to the accuracy and effectiveness of the G-code generated by the simulation in comparison with the actual CNC milling process. This study presents a system revolving around augmented reality (AR) programming to assist in the generation of G-codes for a virtual CNC milling simulation process. The proposed system allows the operator to understand the relation between G-codes and the position of the cutter through AR-based simulation. The system can be used to train novice machinists and observe the milling simulation before the actual machining operations. The G-code generation is based on a tracking and registration algorithm that calculates the related coordinates aided with a heads-mounted-display (HMD). The coordinates are saved on a separate file, which contains the coordinate of the cutter and current material removal rate (MRR), and transferred to a G-code program. The generated codes are then validated with commercially available simulation tools and it was found that only a deviation of no less than 5% exist, proving that the developed methodology is a viable method to substitute conventional means for simulating CNC milling.

KEYWORDS: Augmented Reality, G-Code, and CNC Milling

INTRODUCTION

CNC simulations have been developed in virtual environments for NC tool path verification and machining process optimization [1], but although VR-based systems have already been applied broadly in the manufacturing industry, limitations still exist. Firstly, the system are usually costly, requires powerful hardware, and separates the simulation aspect from the machining aspect, meaning the user has to adjust the experience gathered from the 3D graphic environment to the real machining environment[2]. Secondly, the system is so tightly integrated that it is difficult to support continuous improvement and lessens flexibility, considering that a fully autonomous CNC manufacturing environment from start to finish involves many steps, not just the machining aspect [3]. This brings forward the demand for AR. AR is a rapidly growing field of research that aims to fully integrate virtual with real environment. The development of AR has been around since the early 90s, but has only recently been emerging as one of the forefront of technology, mainly due to the rise of popularity in smart phones and tablets [4]. This proves that AR can be applied in many field of research and even in consumer products due to the lower system requirement being one of the contributing factors. By enhancing the users' understanding and interactions with the manufacturing environment either through real-time information feedback or audio cues, shorter lead time and lower manufacturing costs can be achieved [5]. In addition, providing simulation in a real environment partly removes the time-consuming geometric and kinematic modeling of the machine tools and accessories in the real environment, thus improving the simulation efficiency.

RELATED STUDIES

The field of visualization is stepping into a new era of development, with emerging hardware support by large corporate due to the realization of a promising future associated with this technology. A system that couples automation with visualization is able to project us to the forefront of technology and achieve the realization of a fully digitalized environment envisioned by humans many years ago. Augmented reality is a form of visualization that has been garnering attention for quite some time and is constantly under development. As of this moment, it has been applied in field such as military, industrial, medical, and entertainment industries. Automation, when coupled with visualization, creates an interesting environment. In the industrial field, the use of robotic arm or computer-controlled machines always require a form of programming to teach them the actions that need to be taken depending on the situation. If this task is improved or enhanced with some form of visualization, then it is safe to say that simulating the automated system will yield a higher productivity due to higher immersion and intuitiveness.

Augmented Reality

Augmented reality is a realm between the real environment and complete virtual environment, and in this transition process, AR takes on both properties as well as the best of both worlds. Latest advancements in the field of AR have been studied upon, and issues such as characteristics, registration errors, error reduction, and potential applications were covered [6]. However, most AR usages do not mature beyond the labs, due to technological limitations, social acceptance, and limitations in user interface. To further evaluate this point, tracking in an unprepared environment still proves to be quite challenging, lack of information portrayed for high-level tasks are still prevalent, and persuading an ordinary user to wear a system on them for the technology to function appropriately will still take some time. The author is convinced that these factors most definitely will not stop the advancement of this technology; on the contrary, it will be a driving factor for future researches instead. Key points under AR are the tracking and registration, various types of display systems depending on its application, and its difference with virtual reality (VR).

Tracking and Registration

Two main activities of AR are tracking and registration. Fiducially, or markers in the real environment relative to the camera are tracked to obtain the position and orientation. The tracking values will be utilized in registration to superimpose the 3D virtual object onto the real environment. Considering that registration is still one of the main hurdles when developing a virtual environment, a projective reconstruction technique which composes of embedding and tracking was used as a generalized registration method[7]. Embedding is when four points are specified as reference points to build the world coordinate system and tracking is when 3D coordinates are used to track the four points and calculate the registration matrix. The program is compiled with C++ while the virtual objects are rendered with OpenGL after exporting from CAD commercial software tools. Among the advantages from this proposed method is that there will be no occlusion problem present and real-time adjustments are possible to be conducted. Adding on, a mobile positioning system that is able to track coordinates for indoor construction, then further realize them into three-dimensional positioning with the aid of augmented reality was a focus in a recent study [8]. To achieve this, an infrared marker technique is used, which is invisible and used to develop the Head Marker Tracking Augmented Reality (HMTAR) system. This system allows indoors or outdoors mounting with event memory and scenario sharing mechanism. The overall HMTAR system consists of a mobile display module and a position detection module, executed with the AREyeCenter package. The author demonstrated that the system is able to maneuver over various terrains, and a marker is not necessary when positioning the virtual objects.

CNC Machining

CNC was specifically used for industrial use, though currently it has gradually stepped into education applications to expand the knowledge of operating a CNC machine [9]. This then gives way to virtual machining, which was used for estimations and observations. Virtual CNC machining might be a new concept to some, but several studies have already analyzed its possibility on application in industries, and some have already been applied. A study was made on the application of virtual machining in relation to its structural analysis as well as the challenges faced in the ongoing research in this field of technology [10]. How a machine tool interacts with the cutting tool and the controller will directly affect the resulting CNC process. The technology we have today allows for the prediction of tool collision and path error checking by graphical means, but accuracy still remains a primary concern. These challenges are exactly the reason why despite virtual manufacturing being available for some time, is still unable to achieve near perfection. Existing virtual systems were reviewed in a recent article that covers from VR-based systems, to mathematical modeling and NC based simulation [11]. VR technology is often associated to complex programs, sophisticated headsets, and expensive hardware. This is not exactly completely wrong, but it is a misconception as well. There are two types of VR systems in general, namely immersion VR and desktop VR. Desktop VR is more common, though immersion VR is entering the mainstream market as it is increasingly affordable and more easily available. An unsolved issue until this very day is achieving flawless real-time simulation, even with the aid of web-based technologies. In conclusion, much work is still required as it is a multi-disciplinary task with various approaches.

METHODOLOGY

The study is initiated with basic programming using ARToolKit to create a running program that can generate AR content through a marker based tracking method. The program includes a block of work piece rendered on the reference marker and an end mill imported from a stereo lithography (STL) file rendered on another marker. The kinematics of the milling machine are borrowed from actual CNC milling machine which negates the need for a kinematic modeling while at the same time ensures that there is no error in positioning values since the markers are placed on the CNC machine itself for an in-situ simulation. For a more effective information feedback, the addition of a HUD is done through programming as well, which shows the parameters determined by the user in an output TXT file, as well as well calculated machining parameters like federate, approach length, length of over travel, cutting time, and material removal rate, which are updated live. The save-point feature by mouse operation allows the user to save a desired point anytime during operation, which exports the coordinate data of the cutter tip relative to the work piece coordinate system for G-code generation. These codes can then be transferred to the actual milling machine to reproduce the physical work piece which is the same as the simulated virtual one.

AR Programming

Microsoft Visual C++ 2008 Express Edition is sufficient to act as the program compiler, and ARToolKit v2.72.1 edition will be used as the tracking system. OpenGL will also be utilized to render all virtual objects in the AR environment. Pro-Engineer Wildfire 5 is used to produce 3D cad models of each machine. The validation step is achieved with Master CAM for G-code comparison. Finally, Fraps v3.5.99 is used as a real time video capture software to record the AR environment. Most of these software tools are either free, or available for use in the Department of Design and Manufacture in University of Malaya.

Relative distance between markers is highly dependent on the transformation matrices assigned to each marker and is computed in the program. A simple formulation of distance between two 3D points is the core formula, which can then be associated with any user input method such as keyboard or mouse input. These two points can either be static or variable, as the distance between them can be updated live to calculate the distance at that point of time. This is the simplest form of collision detection and is the basic premise in computing relative distance. For example, if we let the first point be (x_1,y_1,z_1) and the second point is (x_2,y_2,z_2) as shown in Figure 1, then the distance between them is no different from the formulation of distance for 2D points, only with the addition of the z-axis value.

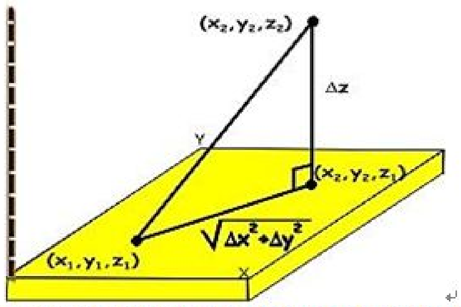


Figure 1: Illustration to calculate distance between two points in 3D space

The distance can be found as

$$Distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \qquad (1)$$

In terms of coding in C++, the code is as stated below:

```
dist = sqrt((position1o-w)*(position1o-w)+(position2o-d)*
        (position2o-d)+(position3o-e)*(position3o-e));
```

Where, the values representing *position1o*, *position2o*, and *position3o* are the coordinates of the tip of the manipulator and is a variable, whereas *w*, *d*, and *e* is the last known position of the workpiece and is considered as a constant.

G-Code Generation

G-code programming has become a standard in operating NC machines since the early 1960s. The code is able to inform the machine and send commands tied to each specific code, usually the movement of the cutting tool according to a tool path. This is just one of the many implementations, as G-code nowadays are used is used as the primary control system for CNC machines. The format of a G-code is made up of several blocks, where each block contains a series of G-codes with specific functions.

A G-code block format is as shown:

```
N100 G01 X1.0 Y2.0 Z3.0 F4
```

Where,

- N100 = No specific meaning, it shows the operator the identification number for each block. A common practice would be to increase the value by an incremental of 5 to allow any additional blocks to be inserted in future or corrections.
- G01 = A common G-code present in almost all NC programs, which represents linear interpolation in the cutter's movement. It moves at a specified feed rate and is usually used during any cutting operations.
- X1.0 = Specifies the X coordinate axis for the cutter to follow. In this example, the code orders the cutter to move 1 unit in the X direction.
- Y2.0 = Specifies the Y coordinate axis for the cutter to follow. In this example, the code orders the cutter to move 2 units in the Y direction.
- Z3.0 = Specifies the Z coordinate axis for the cutter to follow. In this example, the code orders the cutter to move 3 units in the Z direction.
- F4.0 = Informs the NC machine of the current feedrate for linear interpolation. It does not need to be called every time, and depends on the cutter operationThe value is usually brought forward like any other programming method, where if the value does not change, it remains as 4 throughout the program.

Of course, the G-code blocks do not comprise solely as those mentioned above, and a long list of blocks beginning from turning on the machine to shutting down, creates a complete G-code. There are limitations present as well when writing a block, such as a maximum of four G or M words with no repetitions. The X, Y, and Z axis movement follows the work piece coordinate system (WCS) as shown in Figure 2, where the origin is defined by the programmer. Normally, the WCS is set to the top left corner of the work piece. By placing all three axes movement in a single block, the cutter will move to its destination directly in a linear line. If the programmer wishes to move each individual axis separately yet still reach the same destination, it is possible by separating each of them into different blocks and to be called out separately. This means that how each code is placed in a block is important in determining the machining operation sequence.

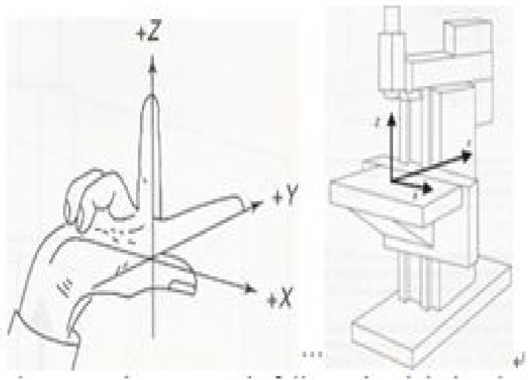
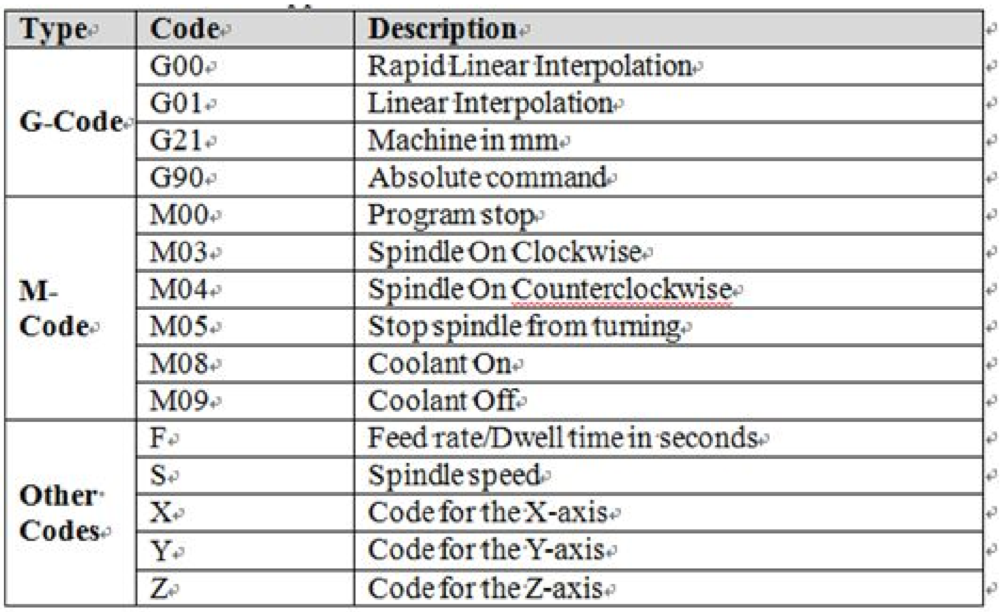


Figure 2: The WCS Axis follows the Right Hand Rule (Left) Where the Z Axis Points Towards the Cutter (Right)

One of the key features of the simulation system is the ability to generate G-code blocks based on the virtual environment and the placement of the cutting tool relative to the WCS. This allows the user who runs this simulation to obtain all the necessary codes in machining a standard operation using a 3-axis vertical milling machine. Despite the system only supporting 3-axis, complex operations can still be carried out, evident by past applications for even non-uniform surfaces like sculpturing with NC machining [12]. Furthermore, extension to 4- and 5-axis CNC machines can be done once 3-axis machining is properly established [13]. The lists of G-codes that are supported are shown in Table 1.

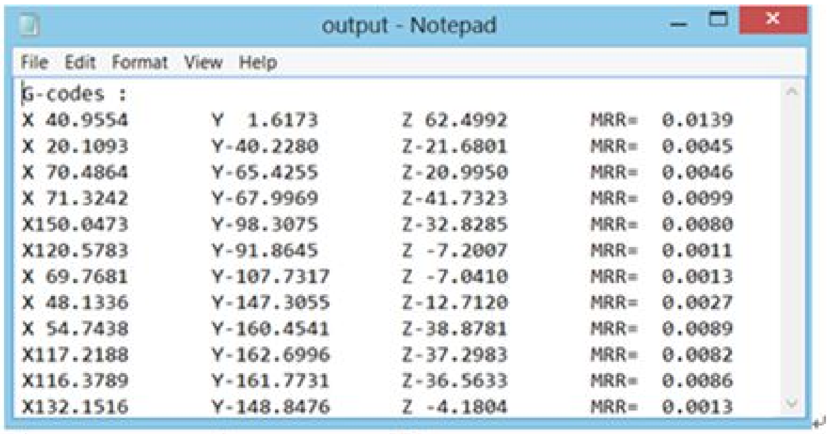
Table 1: List of Code Support



Type	Code	Description
G-Code	G00	Rapid Linear Interpolation
	G01	Linear Interpolation
	G21	Machine in mm
	G90	Absolute command
M-Code	M00	Program stop
	M03	Spindle On Clockwise
	M04	Spindle On Counterclockwise
	M05	Stop spindle from turning
	M08	Coolant On
	M09	Coolant Off
Other Codes	F	Feed rate/Dwell time in seconds
	S	Spindle speed
	X	Code for the X-axis
	Y	Code for the Y-axis
	Z	Code for the Z-axis

Certain functions of the G-code are not generated the same way, as some of them act as toggle switches. G00, G01, M00, M03, M04, M05, M08, and M09 are tied to toggle switches specific to keyboard functions. The values of F and S are calculated under machining parameters. These codes can all be seen on the HUD with visual cues, such as the work piece becoming blue color when the coolant is switched on. Therefore, the key values in a G-code programming, which are the X, Y, and Z, values, are tied to the *save Coordinate* function. This function is specifically designed to operate with the mouse input. *Save Coordinate* means saving the current position when the mouse button is clicked. The *save Coordinate* function is called by placing it in the *mouse Event* function. *GLUT_RIGHT_BUTTON* and *GLUT_DOWN* functions state the condition of the mouse when it is right clicked once. Positions saved are used for the input values and verification to be made in commercial software, Master cam.

In the *save coordinate* function, the coordinates and current MRR are saved to a text file. An example of the text file is shown in Figure 3, which is generated based on a total of 12 mouse clicks. Therefore, each mouse click generates one line of information, and the MRR value depends on the current depth at that point. In reality, saving the coordinate means that each mouse clicks enters the value of the current location of the virtual cutter relative to the origin. This is the reason why the G90 code is used. The G90 absolute command code ensures that every cutter movement is relative to the WCS origin, as shown in Figure 4. Therefore, as long as the actual CNC machine utilizes the G90 code as well, there will be little error present during the programming step.



```
G-codes :
X 40.9554      Y  1.6173      Z 62.4992      MRR=  0.0139
X 20.1093      Y-40.2280      Z-21.6801      MRR=  0.0045
X 70.4864      Y-65.4255      Z-20.9950      MRR=  0.0046
X 71.3242      Y-67.9969      Z-41.7323      MRR=  0.0099
X150.0473      Y-98.3075      Z-32.8285      MRR=  0.0080
X120.5783      Y-91.8645      Z -7.2007      MRR=  0.0011
X 69.7681      Y-107.7317     Z -7.0410      MRR=  0.0013
X 48.1336      Y-147.3055     Z-12.7120      MRR=  0.0027
X 54.7438      Y-160.4541     Z-38.8781      MRR=  0.0089
X117.2188      Y-162.6996     Z-37.2983      MRR=  0.0082
X116.3789      Y-161.7731     Z-36.5633      MRR=  0.0086
X132.1516      Y-148.8476     Z -4.1804      MRR=  0.0013
```

Figure 3: Saved Output Text File

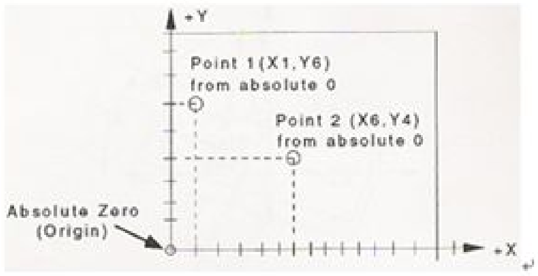

+Y
Point 1 (X1,Y6) from absolute 0
Point 2 (X6,Y4) from absolute 0
Absolute Zero (Origin)
+X

Figure 4: Absolute Programming Where the Origin Acts as the Datum

DISCUSSIONS

To test the simulation system, the milling operation will be conducted in a fashion similar to Master cam, as well as validated with Master cam to determine the accuracy of the operation. Firstly, a CAD model is created using Pro-Engineer with the overall dimensions being the same as the work piece in the simulation. Next, the model is exported as an STL file to be imported into the simulation. The imported model carried over the dimensions of the CAD model, with the same machined slot, as in overlay on top of the original virtual stock. Master cam uses this method as well, where a CAD model is imported into it and superimposed with the stock, or work piece.

In Master cam, the user is required to click each of the edge of the machined part to teach the system on the areas required to be machined, so that the final work piece will look the same as the CAD model. Furthermore, a G-code is generated from the operation. Therefore, for validation purposes, the same procedures are replicated for G-code comparison. With the imported STL model overlaying the virtual work piece, the operator moves the cutter to cut the material according the STL model while saving the necessary coordinates typically at the end of a line. The generated G-code is then compared with the Master cam G-code and any inconsistencies are observed. The setup of the system is shown in Figure 5.

Conducting the case study on a physical milling machine is merely to provide an accurate axis movement for the cutter while at the same time, providing an actual machining or manufacturing environment to the operator. In comparison, the Master cam simulation is shown together with the AR-simulation in Figure 4.4. A key difference that is present in the Master cam simulation is circular interpolation, which is not supported in the AR CNC simulation. At the end of each axial cut, before the cutter path changes to the next axis, it performs a circular interpolation, which creates a variation on the generated G-code.

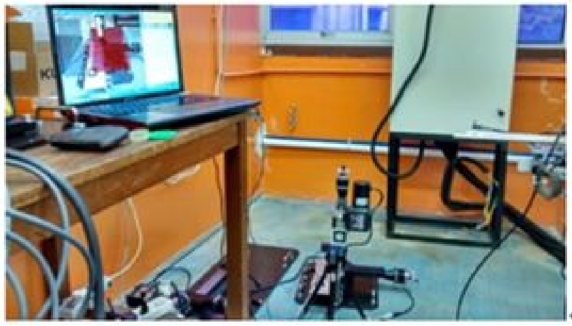


Figure 5: Setup for testing the simulation on Table CNC Machine

The case study aims to validate complex machining that encompasses all three axes values. Typically, a finished product requires machining of at least all three of the axes, therefore this validation is the most accurate representation of a real product simulated. The CAD model is shown in Figure 6.

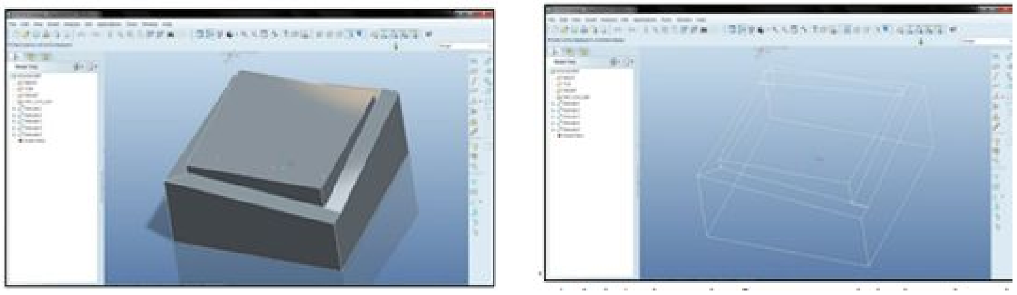
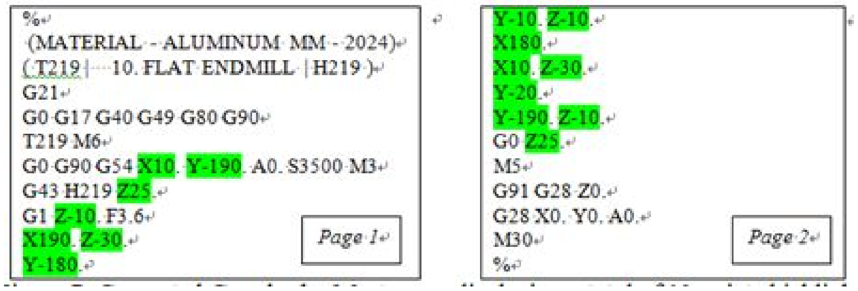


Figure 6: (Left) CAD Model of the Stock (Right) The Wireframe Model Showing the Variable Present for all Three Axes

The cutting is performed on the edges with a slope-like design for a variable depth. The front view shows the top of the slope to be 10mm deep from the surface, as it goes deeper along to the next edge at a depth of 30mm from the surface. The operation then moves to the next edge and back to a depth of 10mm. This cycle goes around the entire edge. It can be seen from the generated code in Figure 7 that Master cam performs a small increment or decrement of 10mm along the X or Y-axis each time it moves to a new edge. Therefore, this same action is replicated on the AR system. Based on the results obtained, the final graph is produced to show the deviation or error for all three axes as shown in Figure 8.



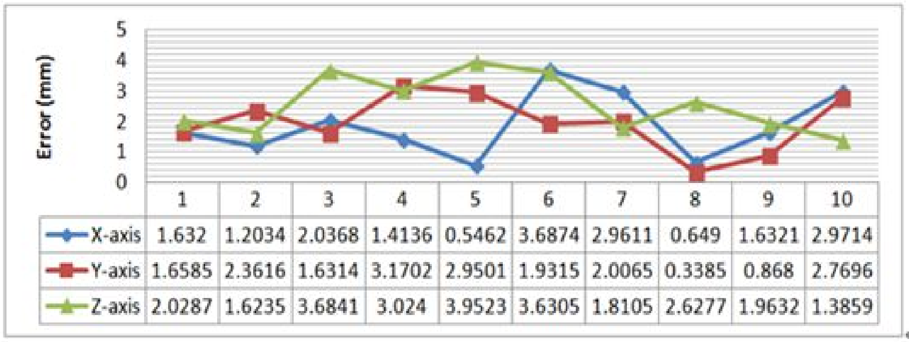
```
%
 (MATERIAL - ALUMINUM MM - 2024)
( T219 |   10. FLAT ENDMILL |H219 )
G21
G0 G17 G40 G49 G80 G90
T219 M6
G0 G90 G54 X10. Y-190. A0. S3500 M3
G43 H219 Z25.
G1 Z-10. F3.6
X190. Z-30.
Y-180.
```

Page 1

```
Y-10. Z-10.
X180.
X10. Z-30.
Y-20.
Y-190. Z-10.
G0 Z25.
M5
G91 G28 Z0.
G28 X0. Y0. A0.
M30
%
```

Page 2

Figure 7: Generated G-Codes by Mater Cam Displaying a Total of 10 Points Highlighted in Green



	1	2	3	4	5	6	7	8	9	10
◆ X-axis	1.632	1.2034	2.0368	1.4136	0.5462	3.6874	2.9611	0.649	1.6321	2.9714
■ Y-axis	1.6585	2.3616	1.6314	3.1702	2.9501	1.9315	2.0065	0.3385	0.868	2.7696
▲ Z-axis	2.0287	1.6235	3.6841	3.024	3.9523	3.6305	1.8105	2.6277	1.9632	1.3859

Figure 8: Error Graph of the 3 Axis Validations

CONCLUSIONS

This study has successfully developed a method for generating G-codes for a virtual 3-axis vertical CNC milling machine which can accurately visualize the process with AR to allow in situ simulation system. This is especially useful to bridge the gap between simulation and actual machining, as one of the key benefits of AR is superimposing the actual environment for a better sense of placement. The user simply needs to operate the machine normally to generate the G-code while observing the virtual work piece being machined according to the desired end product.

This study can be further improved in a number of ways. First and foremost would be the inclusion of a graphical user interface (GUI) for all of the AR systems. Users at the moment need to read the source code or memorize the functions of each keyboard since they are not always shown to the user. Of course, the HUD mitigates the problem partially, but it is overall more intuitive with the presence of a GUI.

ACKNOWLEDGEMENTS

Thank you to the Department of Mechanical Engineering, Faculty of Engineering, University of Malaya, for providing the necessary facilities to support this study. The authors would like to acknowledge for the Ministry of Science, Technology and Innovation (MOSTI) of Malaysia for the financial support under Science Fund, No: 03-01-03-SF0980 (SF007-2014)

REFERENCES

1. Neugebauer, R., Klimant, P. and Witt M. 2012. Realistic Machine Simulation with Virtual Reality. *Procedia CIRP, 2012*. 3(0), 103-108.
2. Zhang, J., Ong, S.K. and Nee, A.Y.C. 2012. Design and Development of an in situ Machining Simulation System using Augmented Reality Technology. *Procedia CIRP*, 2012. 3(0), 185-190.
3. Lin, F.H., Ye, L., Duffy, V.G., and Su, C.J. 2002. Developing Virtual Environments for Industrial Training. *Information Sciences*, 140(1–2), 153-170.
4. Carmigniani, J., Furht, B., Anisetti, M., Ceravolo, P., Damiani, E. and Ivkovic, M. 2011. Augmented Reality Technologies, Systems and Applications. *Multimedia Tools and Applications*, 51(1), 341-377.
5. Nee, A.Y.C., Ong, S.K., Chryssolouris, G., and Mourtzis, D. 2012. Augmented Reality Applications in Design and Manufacturing. *CIRP Annals - Manufacturing Technology*, 61(2), 657-679.

6. Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S. and MacIntyre, B. 2001. Recent Advances in Augmented Reality. *IEEE Computer Graphics and Applications*, 21(6), 34-47.
7. Yuan, M.L., Ong, S.K. and Nee, A.Y.C. 2005. A Generalized Registration Method for Augmented Reality Systems. *Computers & Graphics*, 29(6), 980-997.
8. Kuo, C., Jeng, T. and Yang, I. 2013. An Invisible Head Marker Tracking System for Indoor Mobile Augmented Reality. *Automation in Construction*, 33(0), 104-115.
9. Tseng, A.A., Kolluri, S.P. and Radhakrishnan, P. 1989. A CNC Machining System for Education. *Journal of Manufacturing Systems*, 8(3), 207-214.
10. Altintas, Y., Brecher, C., Weck, M., Witt, S. 2005. Virtual Machine Tool. *CIRP Annals - Manufacturing Technology*, 54(2), 115-138.
11. Kadir, A.A., Xu, X. and Hämmerle, E. 2011. Virtual Machine Tools and Virtual Machining - A Technological Review. *Robotics and Computer-Integrated Manufacturing*, 27(3), 494-508.
12. Inui, M., Kaneda, M. and Kakio, R. 1999, Fast Simulation of Sculptured Surface Milling with 3-Axis NC Machine, *Machining Impossible Shapes*, Springer US. 97-108.
13. Mounayri, H.E., Spence, A.D. and Elbestawi, M.A. 1998. Milling Process Simulation - A Generic Solid Modeller Based Paradigm. *Journal of Manufacturing Science and Engineering*, 120(2), 213-221.