Low-cost concept of tool breakage detection system

1st Martin Kratochvil
Department of Machine Design
University of West Bohemia
Pilsen, Czech Republic
kratochv@kks.zcu.cz

2nd Roman Cermak
Department of Machine Design
University of West Bohemia
Pilsen, Czech Republic
rcermak@kks.zcu.cz

***Abstract*-When machining anisotropic material with a thin milling cutter, there is a high probability of tool breakage. Tool breakage leads to time waste, because a conventional small CNC machine is unable to detect tool breakage. One way to solve this problem could be system based on OpenCV library, which uses a cheap digital camera.**

Keywords -OpenCV processing, CNC mill, breakage detection

I. INTRODUCTION

Thin plywood or fiberglass sheets are used to some extent in the production of prototypes of robots or experimental models. Parts from these sheets can be produced relatively efficiently on small CNC milling machines. Especially in the case of very small details, it is necessary to use cutters with a diameter of 3 mm or smaller. When using such small ones in combination with the material used, the probability of breaking off the cutting part of the tool increases rapidly.

Fig. 1. Comparison of new and broken tool, 1.5 mm diameter

II. PROBLEM DESCRIPTION

A. Milling cutter

Milling cutters of small diameters (up to 4 mm) are made of carbon steel "Fig. 1" or alloy steel. Tool material should be as hard as possible. However, as the hardness of the material increases, the brittleness increases as well. The cutting part is usually formed by two or more helices with a smooth or grooved profile. The blades can be hardened to increase durability. However, in the case of hardened cutters, the core is also hardened due to the thickness of the material, which leads to an increase in the probability of fracture. Small diameter cutters work efficiently at high speeds (> 10000 rpm), the average tool life, especially when processing fiberglass, reaches a maximum of several tens of minutes.

B. Properties of the machined material

Both plywood and fiberglass are not suitable materials for machining, however, their mechanical properties predetermine their use as construction materials. Plywood sheet consists of several layers of veneer joined by a glue. There are natural cracks in the veneer which are filled with glue during production. When machining plywood, the glue dissolves. The dissolved glue clogs the tool edges, which leads to a reduction in chip removal, the consequent increase in lateral forces breaks the tool.

Fiberglass is made of glass fabric filled with epoxyresol resin. When machining fiberglass, the cutting forces fluctuate considerably due to the internal fiberglass structure. This, together with the hardness of fiberglass fabric, results in relatively strong vibrations that can break the tool. In some cases, fiberglass resin is also affected by heat, which is manifested mainly by an increase of friction during machining and tool clogging.

C. Analysis of possible solution

The following variants have been considered to solve the problem of tool breakage:

- Adjustment of cutting conditions and preventive tool change
  Advantage: There is no need to do anything else.
  Disadvantage: The probability of tool breakage is only reduced, productivity is reduced and (variable) tool costs increase. The resin problem is not solved.
- Checking by passing the tool through a light beam (laser)
  Advantage: Possibility to asses tool wear
  Disadvantage: Requires integration into the machine and control system, practically unfeasible for cheap machines for economic reasons.
- Detection of a change in the vibration spectrum of the workpiece / spindle / machine [1]
  Advantage: Relatively simple sensor installation.
  Disadvantage: The accelerometer must be mounted on a frame or workpiece. However, the spectrum of vibration is affected by many factors could result to difficult evaluation of the fracture.
- Checking the shape / size of the tool using machine vision [2]
  Advantage: cheap method, tool can be checked by an operator on PC screen too.
  Disadvantage: Requires a sufficiently small camera placed close to the tool, possibly supplemented by a system of mirrors and lighting.
- Measurement of the amount / existence of chips in the exhaust
  Advantage: Photoelectric sensor could be small and durable.
  Disadvantage: Small tools produce a minimum of chips, the amount of chips varies depending on the type of operation, the ingress of dirt from the outside environment.

D. Required properties of the solution

Due to the required application, the decisive factors are:

2021 International Conference on Applied Electronics (AE) | 978-80-261-0973-0/21/$31.00 ©202110.23919/AE51540.2021.9542917

ISBN 978-80-261-0973-0, © University of West Bohemia, 2021

- Low cost - use of open-source tools
- Independence of machine construction and control system
- Optional: repairs and modifications by yourself

Due to the above-mentioned requirements, the use of the Python programming language and the OpenCV computer vision library seems to be optimal. Both Python and the OpenCV library are cross-platform, thanks to which they can be run on various platforms (x86, x86-64, ARM, …) and operating systems (Windows, Linux).

III. Description of Proposed Detection System

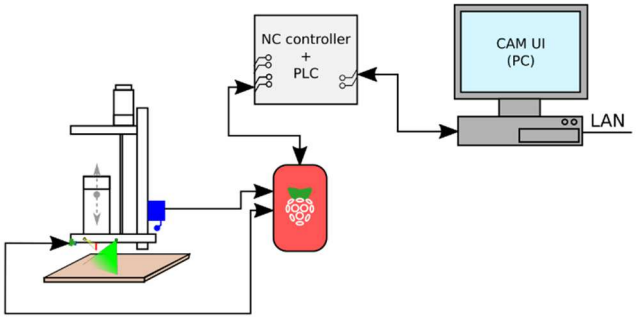

Fig. 2. Scheme of the proposed system

A. HW data processing

The data is being processed using an ARM based computer (Raspberry 3B / 4B) or x86-64 system if higher performance is required. The operating system is Raspbian (open-source preference). A signal is sent via this interface to the PLC module of the machine control system. “Fig. 2”

B. SW data processing

The data processing system is implemented in the Python3 programming language using the OpenCV library, or other suitable libraries (Numpy, SciPy)[3,5]. Independently of the user, the tool will be photographed by the camera, processed and stored in the memory as a reference value. During machining, shooting and evaluation takes place at each pass, the command is issued by closing the switch on the spindle frame. If there is a certain deviation from the reference value, a command is issued to interrupt the machining program (pause).

C. Tool scanning

The command to scan the tool should be issued while crossing the board and the tool is not covered. A reference switch should be used for this, which closes the circuit when the spindle frame is being positioned for the next step. As a result, it is not necessary to take machining status data from the machine control system.

D. Sensor

Required sensor is a CCD / CMOS camera with a resolution of at least 640x480, theoretically a monochrome is sufficient, color picture would however simplify the recognition of the tool from the background. The camera must be connectable to a computer via USB or an RJ-45 connector. An important criterion is the existence of a Linux driver, the ideal state is the ability to control the camera in Python3 or CLI. Due to mounting on the spindle frame, the camera must be vibration resistant. “Fig. 3”

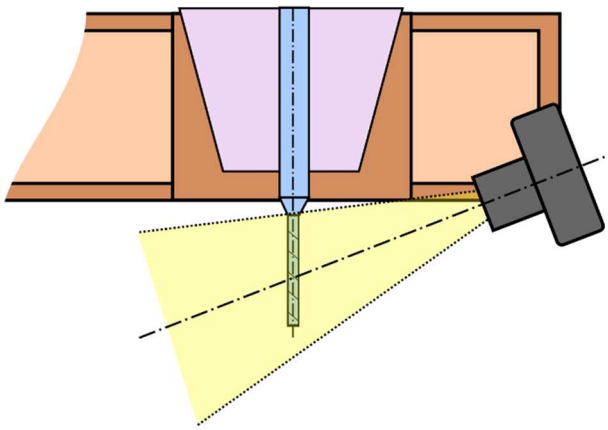

Fig. 3. Proposed position of the sensor, FOV yellowed

E. Increasing the sensitivity of detection

When using a monochrome camera, objects behind the tool should be highlighted. When using a color camera, the instrument and background can be illuminated by LEDs with a different color spectrum. Although the cutter is made of steel and is likely to be gray in the image, the backlight can affect the color of the material, which can increase the difference in pixel values.

F. User/operator interaction

Program interruption should be announced by switching a warning light on, via cellular network or an e-mail.

IV. Description of Image Processing

A. Solution prerequsites

When developing a program, it is advisable to use the following features:

- The tool has a cylindrical shape and is slim – detection of a rectangular object is needed.
- The tool breaks in the neck - the ratio of L/D under the neck is usually greater than 6. At least the presence of the object must be detected. This fact is important, detecting the presence of an object is significantly easier than detecting edge wear in large tools.
- Camera – due to the price, a cheap webcam with USB interface is used. The camera has low resolution (photo 1280x720, video 720p and 640x480). As a rule, these cameras have a small chip, the image is degraded by noise, and there is a decrease in frame rate due to automatic increase in exposure time in low light conditions.
- There is not enough light under the machine spindle – the workspace must be illuminated by LEDs positioned in such a way that the light facilitates tool detection.

The algorithm consists of several consecutive steps, the goal is to modify the captured image to a form that allows the extraction of edges. Proposed solution procedure:

1. loading image

2. preparation for processing
3. edge detection
4. filtering lines
5. deciding whether break-off has been detected

B. Image input

The suitability of using OpenCV can already be demonstrated by the way the camera and Python run environment are connected. The *cv::VideoCapture* class (in Python *cv2.VideoCapture*) is used to connect the webcam. VideoCapture includes an API that captures images from the camera. If the camera identifies the operating system, the image can be loaded using the *.read()* method without additional action. [3]

Fig. 4. Picture of the tool captured by the camera

C. Preparing an image for processing

The next step is to prepare the image "Fig. 4" for processing. The goal of this step is to reduce the amount of image data over which lines will be inserted. Due to the nature of the environment (gray to black tool, color background, increasing contrast between tool and material suitable lighting), edges will be extracted from the image. Edge processing is simplest on a black-and-white image (or 2D array). The advantage of a color camera is primarily the possibility of clearing the background, which can be done by creating a mask corresponding to the background color and a logical *bitwise_and*. From a computer vision point of view, the edge is located where there is a minimum difference in the brightness value between adjacent pixels. To eliminate errors, edge evaluation is performed on a black-and-white image (grayscale). The image "Fig. 5" is converted to the black-and-white spectrum by the converter *COLOR_BGR2GRAY*.

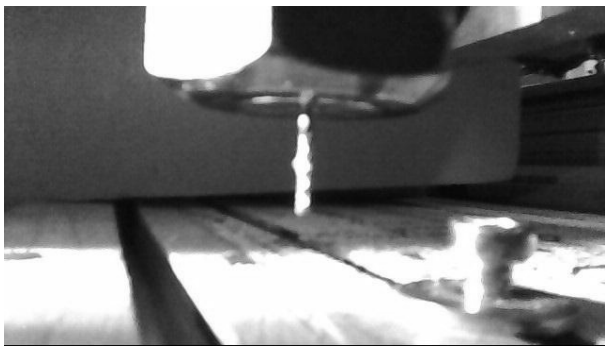

Fig. 5. Black and white image

D. Reduction of unevenness on the cutter image

Certain types of cutters have very rough edges. If the cutter stands or rotates at a very low speed, unevenness and irregular reflections are noticeable in the picture. Then the extracted edges are uneven and cannot be striped through lines. Edge unevenness is eliminated by Gauss blur. First, two images are created, each of them is focused by the *cv2.GaussianBlur* only in one direction, *x* or *y*. Subsequently, the two images are interposed by *cv2.add*. Although the result "Fig. 6" is a relatively blurred image too, when the edges are detected, the effect of blur can be sufficiently eliminated by setting a threshold.

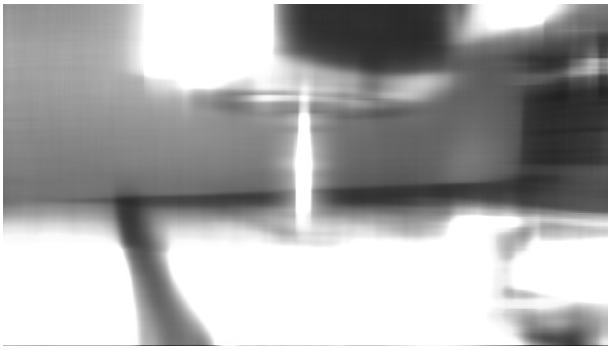

Fig. 6. Black-and-white image cleaned of tool unevenness

E. Edge extraction

Edge extraction is performed by two algorithms – Canny edge (Canny's edge detector) and additional Gaussian pyramid.

1) Canny Edge

Canny edge is edge detection working on the principle of local maximum gradient size. In OpenCV, the entire algorithm is implemented by *cv2.Canny.* The input of the function is a grayscale image. The output is a black-and-white (BW) image where the edges are marked with white pixels "Fig. 7". The output of the function can be modified by operands (OpenCV Docs).

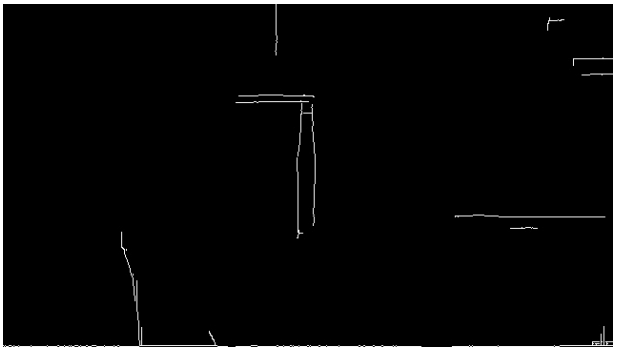

Fig. 7. Edge detection using Canny Edge

2) Gaussian pyramid

During testing, edges were not detected under certain lighting conditions. For this reason, the program was supplemented by edge detection using the Gaussian pyramid[4]. Gaussian pyramid was created by the difference (*cv2.subtract*) between the values of the grayscale image and the image modified by Laplacian pyramid "Fig. 8". Compared to the Canny edge (relatively continuous edges), using the Gaussian pyramid algorithm is less accurate in this way, the image contains

more noise. For this reason, the position of the calculated lines may differ slightly from the actual lines.

Fig. 8. Edge detection using Gauss's pyramid

F. Edge aproximation by lines

Not all detected edges correspond to the edges of the object being monitored. The tool will have the shape of a rectangle in the image, with a greater camera slope a four-wall symmetrical according to the tool's line. Edge aproximation can be realized by using the Hough transformation. In OpenCV, the line detector is implemented by functions *cv2.HoughLines* and c*v2.HoughLinesP.* The program has used the function c*v2.HoughLinesP*, output of which are the coordinates of the *x* and *y* endpoints "Fig. 10". This feature allows to make line detection conditional on their minimum length or maximum edge break value (gap).

G. Selection of lines matching the tool shape

Despite the limitations given, for example, by the minimum required line length, many additional lines are created in practice that do not match the edges of the tool. Lines are selected "Fig. 9" based on the following criteria:

- The line has a minimum length
- The line should be vertical (with tolerance)
- The line should be in the area of interest (where the toothed part of the tool is located in the image)

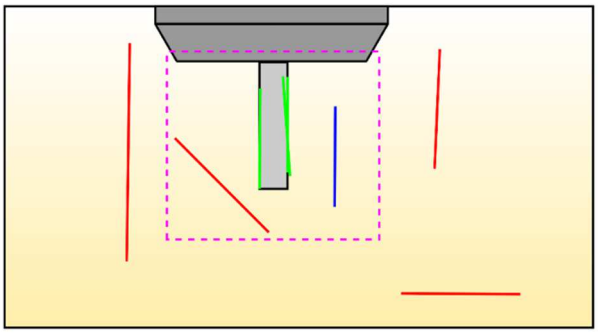

Fig. 9. The principle of selecting lines. Purple area of interest, green line on tool, blue false detection, red lines outside the area of interest.

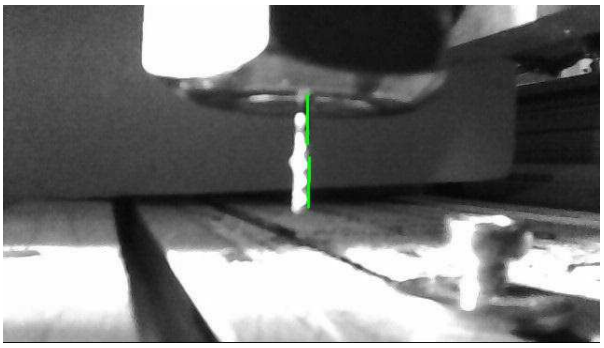

Fig. 10. Plot lines in the tool area

H. Determination of the break-off point

Detection based on Computer vision probably does not have a 100% success rate. In order to evaluate the break-off point, it is necessary to determine, based on the test phase, how many percent of cases both the failed detection of the unbroken tool and the false detection of the already broken tool caused by the background will not occur. Depending on the evaluation of the test phase, the expected solution to this problem is either to exceed the threshold number of consecutive cases where the tool has not been detected, or to score results and interrupt when the specified limit is reached. The scoring system is likely to be more appropriate, as a single false detection would reset the counter of consecutive tool breakages.

V. Conclusion

The aim of this contribution was to explore the possibilities of tool breakage detection using the OpenCV system implemented in Python. The work described the motivation to solve the problem, analysis of detection possibilities and system design. The main part is the description of the algorithm, which can detect whether the working part of the tool has been broken off. The result of the work is a simple prototype program by which it is possible to detect the presence of the active part of the tool in the image captured by the camera. The final program will be able to be introduced after performing the planned tests, during which it will be necessary to determine the parameter settings of some functions and probably adjust the algorithm so that the success rate of detection of the tool has been increased.

Acknowledgment

This research is supported by project SGS-2019-001.

References

[1] S. Sun, X. Hu, and W. Zhang, "Detection of tool breakage during milling process through acoustic emission," Int J Adv Manuf Technol, vol. 109, no. 5–6, pp. 1409–1418, Jul. 2020, doi: 10.1007/s00170-020-05751-7.

[2] J. Yu, X. Cheng, L. Lu, and B. Wu, "A machine vision method for measurement of machining tool wear," Measurement, vol. 182, p. 109683, Sep. 2021, doi: 10.1016/j.measurement.2021.109683.

[3] OpenCV team. *Online Documentation*. Accessed: Feb.20.2020. [Online]. Available: docs.opencv.org/3.4/

[4] Read the Docs, Inc & contributors. *OpenCV-Python Tutorials.* Accessed: Feb.20.2020. [Online]. Available: opencv-python-tutroals.readthedocs.io/en/latest/

[5] Numpy project. *Documentation*. Accessed: Feb.20.2020. [Online]. Available: numpy.org/doc/1.18/reference/index.html

[6] Python Software Foundation. *Docs*. Accessed: Feb.20.2020. [Online]. Available: docs.python.org/3/.