COMPUTER-AIDED DETECTION OF DEFECTS IN THE INVESTMENT CASTING

Krzysztof ZABA, Sandra PUCHLERSKA, Jaroslaw PYZIK

AGH University of Science and Technology, Krakow, Poland, EU,
krzyzaba@agh.edu.pl, spuchler@agh.edu.pl, jaro.pyzik@gmail.com

Abstract

Computer-aided image recognition methods are non-invasive, easy to implement and quick to calculate defects detection methods. They seem to be a promising method for investment casting applications - defects can be detected in individual incestment casting processes, reducing the costs caused by defective castings.

As part of the research, defects have been defined and described in wax models. For each of the disadvantages, a characteristic signature was created allowing for its later detection in the image. In the next stage pre-processing of models was carried out, including segmentation, denoising and sharpening in order to prepare images for the input form for the algorithm. Next, an algorithm for searching and classifying areas containing separate defects and deviations from correct images was developed. The algorithm uses statistical classification methods and machine learning elements using convolutional neural networks.

Keywords: Investment casting, wax model, image recognition, neural networks

1. INTRODUCTION

The most important benefits of the investment casting process are superior surface finish, complex geometries and fine details of the products, the possibility of using wide range of alloys. To create the cast with this method, first we need to make wax model by injection to the metal die. The next stages are mounting models to the sprue, building a ceramic shell around the wax tree, burning off the wax and metal pouring. From the first stage of production, the quality of future castings depends a lot. Wax model is replica pattern of the cast. Making wax models can generate a lot of defects [1], the main along with the causes are:

- deformation of wax pattern - may occur due to improper work of workers or by wrong cooling of the wax
- short - too low wax temperature
- surface wrinkles - insufficient pressure during injection or inappropriate temperature of the wax
- flash - inappropriate fastening of the die
- cracking - effect of mechanical action
- bubbles - can appear when air is injected into the mold along with the wax

The research carried out as part of this article aims to develop algorithms for the visual recognition of defects in wax models. It will be one of the elements of a dedicated system for defect identification in investment casting. Image recognition, also known as computer vision, can be generally described as converting the image by suitable devices or machines to a digital form for further processing. **Figure 1** shows a diagram of a standard procedure in the analysis and image processing.

Figure 1 Image processing algorithm [2]

Often this type of processing is implemented using computational intelligence methods [3]. The authors used neural networks as one of the methods of images analysis and recognition. Neural network is a kind of computational model that operates in the same way as neurons in the human brain. Neuron takes input data, performs operations, and then transmits the output data to the next neuron. Schematic operation is shown in **Figure 2**.

In the frame of the article pre-processing of wax models was carried out, including segmentation, denoising and sharpening in order to prepare images for the input form for the algorithm. The neural network of an image processing based algorithms were implemented to detect different types of defects visible on wax models images.

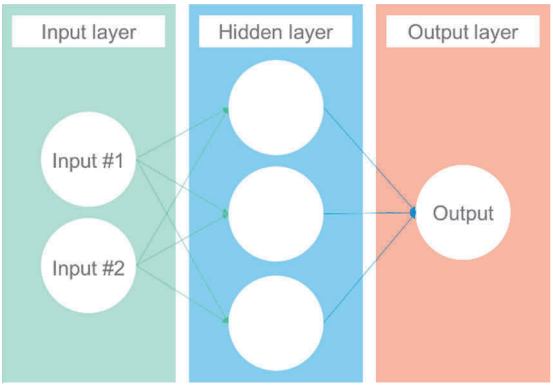

Figure 2 Schematic representation of neural network [4]

2. METHODOLOGY

The purpose of the research was to implement defect detectors for common defects occurring in wax models, based on wax model image. Considered defects were: non-fills, flashes, cracks, flow lines, and graining. For these defects, separate detectors were developed, and complied into main image processing chain. General algorithm for defect detection was as follows (**Figure 3**).

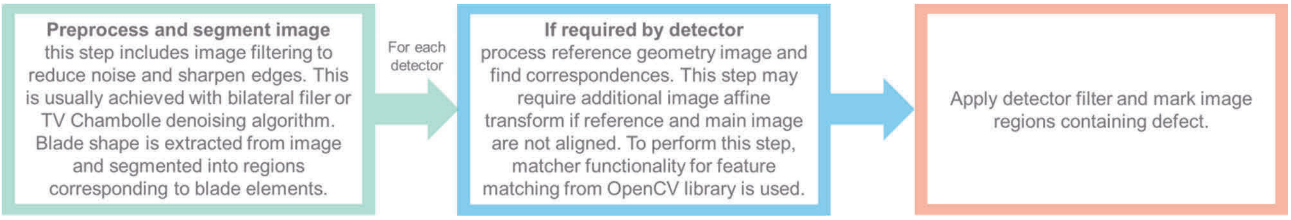

Figure 3 The algorithm for defect detection

Initial step of defect detection process was blade extraction from images and segmentation into regions corresponding to blade base, lateral surface and blades upper part. Obtained segmentations allowed us to analyze important regions of blades separately, which helped to increase performance and improved feature detection by reducing effect of geometry changes in wax models. Segmentations were generated with Watershed algorithm, applied to the gradient of image smoothed with bilateral filter (**Figure 4**).

a) b) c)

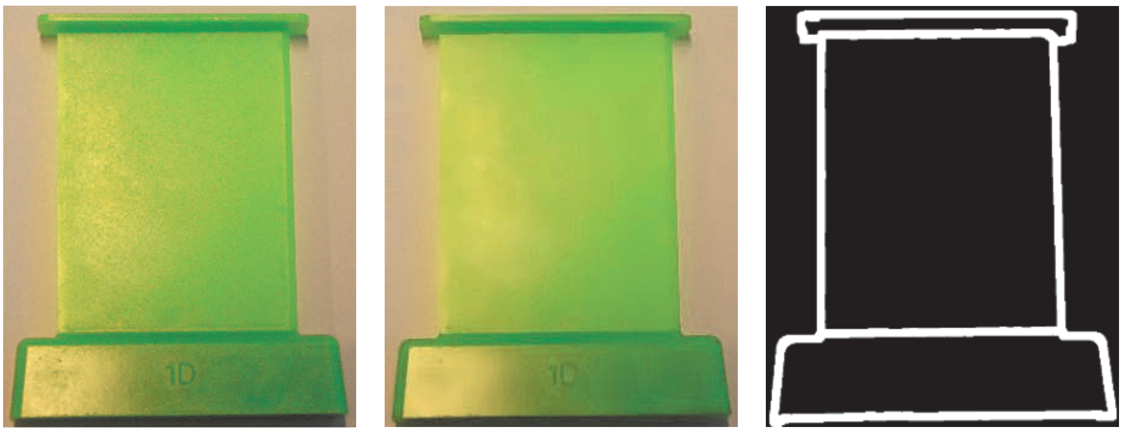

Figure 4 Example original image (a), smoothed image (b), and contours of segmented image (c)

In the next stage of the research, appropriate detectors were used for each of the wax models defects.

3. RESULTS

3.1. Non-fills and flashes

Non-fill and flash defects (**Figure 5**), as deviation from reference geometry, can be detected by taking difference of binary base shape image and binary shape image of model being examined. With given pixel tolerance around shape borders (to reduce image noise and compression effects), and after filtering out small regions, positive pixel values indicate non-fill defect and negative values indicate flash defect. Described approach was implemented for non-fill and flash recognition. This detector requires exact matching with base geometry image. For this purpose, feature matching algorithms from OpenCV [5] library was utilized. In the first step, keypoint descriptors for model image and reference image are generated with SIFT algorithm. Keypoints (features) are characteristic scale and affine transform invariant points of an image like corners or edges. Next step was to find corresponding keypoints on the images with FLANN algorithm and to compute affine transformation parameters based on the keypoints relations. In the final step transformation was reversed, resulting in aligned images that could be processed by the detector.

a) b) c) d) e)

Figure 5 Non-fill defect (a), binary image (b), binary reference geometry image (c), difference of images indicating non-fill defect (d), difference bounding box applied to original image (e)

3.2. Cracks

Crack detection (**Figure 6**) was implemented with image processing routines. First, image was converted to grayscale and adaptive histogram equalization was applied, using CLAHE (Contrast Limited Adaptive Histogram Equalization) algorithm. This allowed to adjust contrast and enhance edges. Then, bilateral filter was used to reduce noise and visibility of surface roughness while preserving edges, which is important for cracks detection. As a final step, Sobel edge detector and binarization was used to extract crack's edges.

a) b) c)

Figure 6 Fragment of blade image with a crack (a), denoised image with edge enhancement (b), and crack detected with Sobel and binarization filter

3.3. Flow lines

For flow lines detection, 3-layer neural network was implemented (**Figure 7**), with 784 input neurons, corresponding to number of input pixels, hidden layer consisting of 400 neurons, and output layer with 2 neurons, indicating if given tile contains flowline characteristic structure. For network learning, we used backpropagation scheme with gradient descent method. As an input, 28x28 tiles were generated from images of blades having flow lines defects. Firstly, images were sharpened with unsharp mask filter, then difference of gaussians (DoG) filter was applied, followed by binarization. In the next step, regions containing flow lines defects, and defect free regions were sampled with overlapping 28x28 windows, with offset of 4 pixels. Obtained tiles were classified as faulty if given tile was covering more than 35% of region with defect, and as defect free otherwise. Based on initial images of blades with flow lines defects, this approach allowed to generate training set of about 19000 images. Different network parameters (such as number of layers and number of neurons per layer) and learning parameters (number of iterations, learning rate) were tested with the best results being yield for values mentioned above.

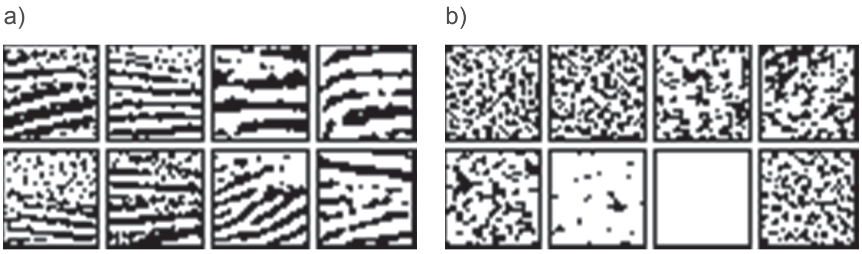

Figure 7 Example training 28x28 tiles generated from images of flow lines defects. Flow lines patterns (a), and regular patterns (b)

After the training, network was tested on blade images, detecting flow lines with 89% accuracy, with incorrect classifications being mainly false positives. To detect defect with trained network, images were processed similarly as in training set generation process - sharpening, DoG and binarization filters were applied. Then, for every image, 28x28 samples were generated and evaluated with trained network. Overlapping regions, classified as regions containing defects were merged, creating final region containing flow lines (**Figure 8**).

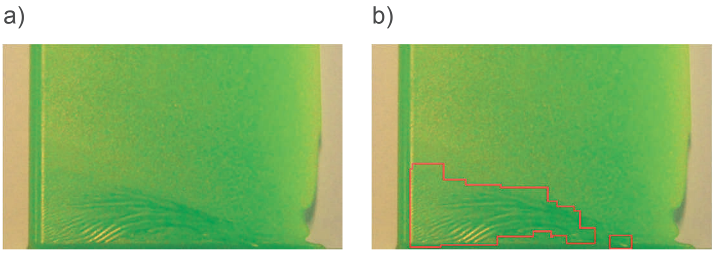

Figure 8 Example input image (a) and flow lines region detected by neural network, applied to original image (b)

3.4. Graining

Graining defect (**Figure 9**) appears on the image as accumulation of darker pixels, creating spare darker regions. Granularity itself cannot be the only measure as image noise can affect recognition and defect free regions also exposure grainy structure with standard enhancing filters. For this reason, to recognize graining effect, median blur, combined with adaptive binarization and morphological opening operation was utilized. In defect regions, median blur would generate continuous, darker regions that can be extracted with binarization routine. Morphology opening was used to filter out small regions.

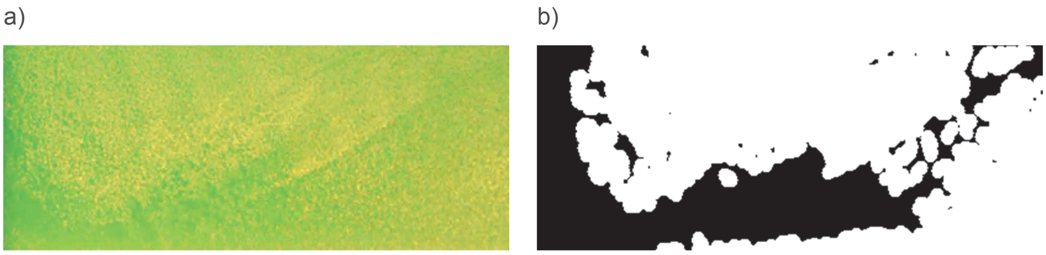

Figure 9 Example input image (a) and flow lines region detected by neural network, applied

Described procedures were implemented as part of a library for defect analysis. Implementation was written in Python language, with use of OpenCV Python [6] bindings and scikit-image modules for image processing. For computational optimization purposes, Theano module was used, allowing also for GPU support for heavy parallel computations.

4. CONCLUSION

With implemented algorithms and routines, good accuracy was achieved, allowing for quick assessment of wax models, based on models' images. Methods of detecting and extracting visual characteristics of defects from images were implemented for non-fills, flashes, cracks and flow line defects. These methods however, are sensitive to image quality, and for industrial application, environment providing reproducible data, consistent background, lighting and photographed object alignment is required to avoid excessive preprocessing and accuracy issues. Neural network implementation was efficient in recognizing flow line defects, with success rate of 89%. Use of convolution neural network is under development, which will improve feature detection, including also described detectors already based on standard image processing routines. This implementation is part of a system for automatic control of quality of intermediate products of investment casting process. Performance, at the library level, is good enough for real time applications.

ACKNOWLEDGEMENTS

The work was done in the framework of a research project No. PBS3/A5/54/2016 funded by the National Centre for Research and Development.

REFERENCES

[1] BHARDWAJ, A., MUKHERJEE, C. K. Wax Defects in Investment Casting and it's Minimization. *International Journal on Recent and Innovation Trends in Computing and Communication.* 2014. vol. 2, iss. 9, pp. 2596-2598.

[2] ZACKOVA, T., JARVELA, I.Y., MARDESIC T. The Role of 3D Ultrasound in Assessment of Endometrial Receptivity and Follicular Vascularity to Predict the Quality Oocyte. *Ultrasound Imaging - Medical Applications, chapter 5,* 2011, pp. 73-88.

[3] TADEUSIEWICZ, R., KOROHODA, P. *Komputerowa analiza i przetwarzanie obrazów.* Wydawnictwo Fundacji Postępu Telekomunikacji, Kraków, 1997. p. 272.

[4] GUILHERME, F. Convolutional Neural Networks. An Overview. *CPQD* [online]. Version 1, Last updated: 27 October 2016. Polis de Tecnologia. 68 p. [viewed 2018-05-08]. Available from: http://polisdetecnologia.com.br/wp-content/uploads/2016/10/CNN_Hyperlink.pdf

[5] BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*. 2000. vol. 25, no. 11, pp. 120-125.

[6] VAN DER WALT, S., SCHÖNBERGER, J. L., NUNEZ-IGLESIAS, J., BOULOGNE, F., WARNER, J. D., YAGER, N., GOUILLART, E., YU, T. Scikit-image: Image processing in Python. *PeerJ*. 2014.