Surface Defect Categorization of Imperfections in High Precision Automotive Iron Foundries using Best Crossing Line Profile

Iker Pastor-López, Igor Santos, and Pablo G. Bringas
S^{3}Lab, DeustoTech Computing
University of Deusto
Avenida de las Universidades 24, 48007, Bilbao, Spain
{iker.pastor, isantos, pablo.garcia.bringas}@deusto.es

***Abstract*—Iron casting production is a very important industry that supplies critical products to other key sectors of the economy. In order to assure the quality of the final product, the castings are subject to strict safety controls. One of the most common flaws is the appearance of defects on the surface. In particular, our work focuses on three of the most typical defects in iron foundries: inclusions, cold laps and misruns. We propose a new approach that detects these imperfections on the surface by means of a segmentation method that flags the potential defective regions on the casting and, then, applies machine-learning techniques to classify the regions in correct or in the different types of faults. In this case, we applied BCLP technique. It provides good information to distinguish between edge structures and defects in this kind of images.**

I. Introduction

Foundry process is one of the most relevant indicatives of the progress of a society. Generally, it consists on melting a material and pouring it into a mould where it solidifies into the desired shape. The resulting castings are supplied to key sectors such as aeronautic, automotive, weaponry or naval industries. Therefore, any defect, even the tiniest one, may become fatal when it comes to the foundry process. In order to discard any defective casting, strict safety controls are required to guarantee a certain threshold of quality for the manufactured casting.

In this context, there are many defects that may appear on the surface of the casting. In this paper, we focus on three of the most common surface defects: (i) inclusions, which are little perforations caused by an excess of sand in the mould; (ii) misruns, that appear when not enough material is poured into the mould; and finally, (iii) cold laps, which are produced when part of the melted material is cooled down before the melting is completed.

Currently, the visual inspection and quality assessment is performed by human operators [1]. Albeit people can perform some tasks better than machines, they are slower and get easily exhausted. Besides, operators are hard to find and to maintain in the industry since they require capabilities and learning skills that usually take them long to acquire. There are also cases of boredom that may affect the process.

In some applications, the inspection is critical and dangerous and computer vision can replace more efficiently and without danger [2]. Computer vision has become an important field that can aid to the visual inspection in quality control processes.

In this sense, computer vision systems are replacing manual inspection in many industries such as timber [3], textile [4] or metallurgical [5] [6]. Whereas manual inspection strongly depends on human factor, computer vision is independent, with the subsequent decrease in the error rate.

Against this background, we propose a new approach capable of detecting and categorising inclusions, cold laps and misruns. First, we describe a machine vision system that retrieves the information of the surface of the tested castings. Second, a segmentation method, based on modeling the correct castings, is used in order to detect the possible defects. Finally, we employ several features extracted from the machine-vision and segmentation systems to train machine learning algorithms to categorise the possible defects.

Summarising, our main contributions are: (i) an adaptation of a machine vision system to the segmentation of foundry casting regions, (ii) a machine-learning approach to categorise faulty regions on the foundry castings and (iii) an empirical validation using actual foundry castings of our proposed approach.

The remainder of this paper is organized as follows. Section II describes how image data is gathered and prepared for next steps. Section III describes the segmentation process that identifies the potentially faulty regions of the images. Section IV details the feature set for the categorization of the different defects. Section V describes the experiments performed and presents results. Finally, Section VI concludes the paper and outlines avenues for future work.

II. Data gathering and Normalization

To gather the superficial data from the castings, we utilized laser triangulation camera. Through the high-power laser (3-B class), our method scans the castings, regardless their dark and uneven surface.

To automate the process of casting scanning, we have used a robotic arm. This automation is only performed to automatically scan the casting. The positioning of the casting on the scanning table is manually performed because the geometric differences between the different casting models we are scanning. Specifically, the robotic arm is a Motoman NH6-10. For the positioning of the casting, we used silicone molds for each casting model.

In order to retrieve the information about the scanned casting, the casting is put on the mold and the scanning is started. The robotic arm performs a linear movement, gathering a set of profiles based on the generated triangulation of the laser and the optical sensor. The surface of a foundry casting $\mathcal{S}$ is a set of profiles $\mathcal{P}$ such as $\mathcal{S} = \{\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_{n-1}, \mathcal{P}_n\}$. Each profile is retrieved for a thickness of 0.2 mm. Each of these profiles are composed of the heights of each of the scanned points. If we join them altogether, we can form a height matrix $A_{m\times n}$ that represents the scanned surface S, such as:

$$\mathcal{A}_{m\times n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n-1} & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n-1} & a_{2,n} \\ \vdots & & \vdots & & \vdots \\ a_{m-1,1} & a_{m-1,2} & \dots & a_{m-1,n-1} & a_{m-1,n} \\ a_{m,1} & a_{m,2} & \dots & a_{m,n-1} & a_{m,n} \end{pmatrix} \tag{1}$$

where each item $a_{x,y}$ represents the height in the space of a given point (x, y) and the dimensions m and n are determined by the size of the scanned casting.

Once the height matrix $A_{m\times n}$ is formed, we first remove the point that represent the scanning table surface, denoted $\mathcal{T}_{o\times p}$ being $0 < o \leq m$ y $0 < p \leq n$. To this end, we set a height threshold, determined by the maximum height of the point known to be part of the scanning table, such as:

$$\sigma = \underset{t_{x,y}\in\mathcal{T}_{o\times p}}{\operatorname{argmax}}\ t_{x,y} \tag{2}$$

being x and y the position of the element a in the scanning table matrix $\mathcal{T}_{o\times p}$.

Once, the threshold σ is set, we subtract this value to each point the height matrix, setting a minimum height of 0, such as:

$$\forall a_{x,y} \in \mathcal{A}_{m\times n} : ((a_{x,y} \triangleq a_{x,y} - \sigma) \mid a_{x,y} \geq 0) \tag{3}$$

As a result, our method removes every data devoid of information about the scanned casting and, therefore, our method generates a new matrix $\mathcal{B}_{q\times r}$, where q is the number of rows whose elements are different to zero within the initial matrix $\mathcal{A}_{m\times n}$ and r is the number of columns whose elements are different to zero within the initial matrix and thus, $0 < q \leq m$ and $0 < r \leq n$.

Hereafter and after our method finish scanning every casting in our dataset, our method has to normalize the different height matrices ir order to guarantee that every matrix has the same dimensions p and q. To this end, we start with a casting dataset $\mathcal{C}$ formed by ℓ matrices $\mathcal{B}_{q_i,r_i}$ where q_i y r_i are the height and width of the i-th matrix in the dataset $\mathcal{C}$. To normalize the matrices in $\mathcal{C}$, we start determining the new dimensions they will have. These are the maximum width and height, , q' y r', such as:

$$q' = \underset{q_i}{\operatorname{argmax}}\ \mathcal{B}_{q_i\times r_i} \in \mathcal{C} \tag{4}$$

where q' is the maximum height of all the matrices $\mathcal{B}_{q_i\times r_i}$ within $\mathcal{C}$. The maximum width is computed the same way:

$$r' = \underset{r_i}{\operatorname{argmax}}\ \mathcal{B}_{q_i\times r_i} \in \mathcal{C} \tag{5}$$

Next, each of the matrices are re-dimensioned with the new width q' and the new height r', assigning the new values:

$$\forall \mathcal{B}_{q_i\times r_i} \in \mathcal{C} : \mathcal{B'}_{q'\times r'} \triangleq align(\mathcal{B}_{q_i\times r_i}, q', r') \tag{6}$$

where $align$ is a function that moves each of the values of each matrix $\mathcal{B}_{q_i\times r_i}$ to the new normalized matrix, denoted by $\mathcal{B}'_{q'\times r'}$:

$$align(\mathcal{B}_{q\times r}, q', r') = B'_{q'\times r'} = \begin{pmatrix} b'_{1,1} & \dots & b'_{1,r'} \\ b'_{2,1} & \dots & b'_{2,r'} \\ \vdots & \vdots & \vdots \\ b'_{q'-1,1} & \dots & b'_{q'-1,r'} \\ b'_{q',1} & \dots & b'_{q',r'} \end{pmatrix} \tag{7}$$

where each element in the new matrix $b'_{i,j} \in B'_{q'\times r'}$ represent the elements in the matrix $\mathcal{B}_{q\times r}$, moved to their new position in the new matrix $B'_{q'\times r'}$, such as:

$$\forall b_{i,j} \in \mathcal{B}_{q\times r} : b'_{i+(q'/2)-(q/2),j+(r'/2)-(r/2)} \in \mathcal{B}'_{q'\times r'} \triangleq b_{i,j} \tag{8}$$

aligning every matrix and normalizing, in this way, their different dimensions, forming a new normalized casting dataset $\mathcal{C}'$, composed of the normalized matrices.

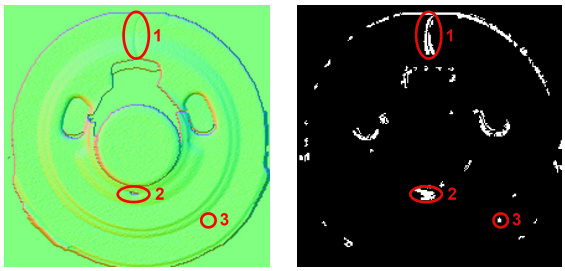
Once the matrices are normalized, our method generates different data representations of the gathered information. In particular, 3 different representations are used:

- *Height Map*. This matrix is the normalized one, our system retrieved from the normalization process.
- *Grayscale Height Map* [7], a well-known representation converts each height value to a range between 0 and 255, showing the different scales of gray.
- *Normal map*. This representation has been generated by means of the height matrix, but shows the direction of the normal vector of the surface for each point in the matrix and each vector for each point have three components (x, y, z) that we represent as an image — even when it is not a true image by itself — corresponding red component to the x value, green component to the y value, and blue component to the z value.

III. Image Segmentation Process

The process of segmentation employs correct castings as models to compare the scanned ones with. Besides these models, we apply different image filters to remove the image noise and improve the information extraction phase.

Depending on the result of the segmentation process, different final steps are applied. In case that the scanned image is determined to be correct, a filter to remove every noise is applied. Otherwise, besides of removing noise, we apply a filter to magnify the detected defects. Figure 1 shows an example of the results of the segmentation for a given image.



(a) Normal map in RGB of the defective casting.

(b) Result image of the segmentation process. The detected defects are marked in red.

Fig. 1. The segmentation process. The fist image shows the representation of the normal map in RGB of a defective castings with 3 defects: 1 for a cold-lap, 2 and 3 two inclusions. The second image show the results of the segmentation process and the detected defects are marked in red.

In particular, the segmentation process is divide in the following steps:

1) Our method converts the original normal map to gray-scale — although it is not an image, it is represented as one, as aforementioned — of the casting and the normal map of the correct models. This step is performed to remove the noise regarding the rugosity on the surface.
2) The Gaussian Blur filter [8] is applied.
3) A difference filter is applied between the normal map to examine and each of the correct models.
4) An intersection filter is applied between the differences computed in the previous step.
5) The method binarizes the results.
6) An algorithm to extract the potentially faulty areas or segments is used, which removes the ones which are excessively small (i.e., regions smaller than 3x3 pixels).

IV. Feature Sets for Categorization of Defects

In this section, we describe the different feature sets that have been extracted from the segmented regions. In this way, we will evaluate which set or sets are the most adequate to categorize them through supervised machine learning.

In particular, the different sets we chose are the following: (i) *simple features*, that correspond with existing information in some of the representations of the regions; and (ii) *BCLP (Best Crossing Line Profile) based features*, that minimize the irrelevant data from the segmented regions.

A. Simple Features

We denote simple features to those that can be extracted directly from the different representations result of the segmentation process and do not require an additional processing. The feature in this category are the following:

- **Features extracted from the binarized image:** In this type of image, the potentially faulty regions are marked in white, whereas the rest are black. We extract the following geometric features of the potentially faulty regions:
  - Height, width, perimeter, and area of the region.
  - Euclidean distance from the gravity center of the region and the origin of the coordinates.
  - Fullness of the region, computed as:

$$Fullness = \frac{Area}{Height \cdot Width} \tag{9}$$

- **Features extracted from the *integral* image, converted from the binarized region:** An integral image I from a base image G is defined as the image where the intensity of a pixel in a specific position is equal to the sum of the intensities of every pixel on top and on the right of the given pixel [7]. We employ the following features:
  - Average value of the pixels of the integral image.
  - Sum of the values of the pixels of the integral image.
- **Features extracted from the height matrix:** In this case, we extract two different regions. In the first one, we gather every pixel in the height matrix regardless of its segmentation result. In the second, we retrieve the pixels marked in white, result of the binarization process. For both data sources, we extracted the following measures: sum, average, standard deviation, standard error, minimum value, maximum value, the difference between the maximum value and minimum value, median, entropy, bias of the pixel distribution, and the kurtosis value of the pixel distribution.
- **Features extracted from the normal vector matrices:** Similarly to the features extracted from the height matrix, we use as data source to compute the measures the unsegmented pixels of the normal vector matrices and potentially faulty pixels of these matrices after the segmentation process is carried out. As in the height matrices, the features extracted are: sum, average, standard deviation, standard error, minimum value, maximum value, the difference between the maximum value and minimum value, median, entropy, bias of the pixel distribution, and the kurtosis value of the pixel distribution.

B. BCLP (Best Crossing Line Profile) based Features

Crossing Line Profile was first proposed by Mery [9] for defect detection in steel castings images obtained through X-Ray scanning. This algorithm is able to detect faulty castings without previous knowledge of the process.

Our method is based in the same concept, but the goal is to extract several features of the castings to train supervised

learning classifiers. To this end, we first resize the segmented region, obtained from the height map of the grayscale image and from the normal vector matrices codified in RGB. In this way, we resize every image to 32x32 pixels, using a nearest neighbor approach [10].

Next, we define a *Cross Line Profile* P_θ, as the function that identifies the line that crosses the region r by its middle point and forms and angle θ with respect to the X axis. In the same way as Mery [9] did, our method computes 8 different profiles P_θ, being $\theta = (K \cdot \pi)/8$ and for each $K \in [0, 7]$.

Once the 8 profiles are computed, our method select the value of K for which the extreme pixels of the CLP profile P_θ are more similar between them. This profile is called the *Best Crossing Line Profile* or BCLP. Through this profile, we can discern between an image border and potential defect of the examined casting. We extract several features from this profile in order to categorize the defects:

- **Features of the BLCP computed from the grayscale height map:** The sum, average, standard deviation, standard error, minimum value, maximum value, the difference between the maximum value and minimum value, median, entropy, bias of the pixel distribution, and the kurtosis value of the pixel distribution are used
- **Features of the BLCP computed from the normal vector matrices codified in RGB:** The sum, average, standard deviation, standard error, minimum value, maximum value, the difference between the maximum value and minimum value, median, entropy, bias of the pixel distribution, and the kurtosis value of the pixel distribution are extracted from the profile.

Co-occurrence matrices is a method that can be applied to measure the textures in an image [11], [12], [13]. In particular, these matrices are computed as follows:

$$CM_{\Delta x,\Delta y}(i,j) = \sum_{m=1}^{k}\sum_{n=1}^{l}\begin{cases} 1, & \text{If } I(m,n) = i \text{ and} \\ & I(m+\Delta x, n+\Delta y) = j \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

where I is an images of size $k \times l$ and Δx and Δy denote the offset. Our method extracts the well-known Haralick textural features [14]: normalization, contrast, dissimilarity, homogeneity, angular second moment, energy, maximum probability, entropy, mean, variance, standard deviation, and correlation.

V. Empirical Validation

To evaluate our surface defect detector, we gathered a dataset from a foundry, specialized in safety and precisions components for the automotive industry (principally, in disk-brake support with a production over 45,000 tons per year). Three different types of defect (i.e., inclusion, cold lap and misrun) were present in the faulty castings. In particular, this dataset is composed of 639 castings: 236 are correct, 392 have inclusion, 8 have cold laps, and 9 have misruns.

By means of this analysis, we constructed a dataset of 6,150 segments to train supervised learning models and determine when a segment is defective. In addition, we added a second category to identify the noise that our segmentation gets called 'Correct', which represents the segments marked by the segmentation method that are correct even though the method has marked them as potentially faulty. In particular, 5,686 were correct and 464 were faulty as shown in Table I.

TABLE I
Distribution of the segmentation process results.

Type	Number of segments
Correct	5,685
Inclusion	392
Cold Lap	17
Misrun	55

To evaluate the precision of the supervised machine-learning methods to categorize the segments. To this extent, by means of the dataset, we conducted the following methodology to evaluate the proposed method:

- **Casting representation:** In order to compare, we represented each potentially defective region using the feature sets described in Section IV. In particular, the next representations were used:
  - Simple features (Simple).
  - Best Crossing Line Profile based features (BCLP).
  - Combination at feature level of Simple and BCLP features (Simple+BCLP).
- **Cross validation:** This method is generally applied in machine-learning evaluation [15]. In our experiments, we performed a K-fold cross validation with $k = 10$. In this way, our dataset is split 10 times into 10 different sets of learning and testing. For each fold, we changed the number of labeled instances from 10% to 90% to measure the effect of the number of previously labeled instances on the final performance of collective classification in detecting surface defects.
- **SMOTE:** The dataset was not balanced for the different classes. To address unbalanced data, we applied Synthetic Minority Over-sampling TEchnique (SMOTE) [16], which is a combination of over-sampling the less populated classes and under-sampling the more populated ones. The over-sampling is performed by creating synthetic minority class examples from each training set. In this way, the classes became more balanced.
- **Learning the model:** For each fold, we accomplished the learning step of each algorithm using different parameters or learning algorithms depending on the specific model. The algorithms use the default parameters in the well-known machine-learning tool WEKA [17]. In particular, we used the following models:
  - *Bayesian networks (BN):* With regards to Bayesian networks, we utilize different structural learning algorithms: K2 [18] and Tree Augmented Naïve (TAN) [19]. Moreover, we also performed experiments with a Naïve Bayes Classifier [15].
  - *Support Vector Machines (SVM):* We performed experiments with a polynomial kernel [20], a normal-

ized polynomial Kernel [21], a Pearson VII function-based universal kernel [22] and a radial basis function (RBF) based kernel [23].
  - *K-nearest neighbor (KNN):* We performed experiments with $k = 1$, $k = 2$, $k = 3$, $k = 4$, and $k = 5$.
  - *Decision Trees (DT):* We performed experiments with J48(the WEKA [17] implementation of the *C4.5* algorithm [24]) and Random Forest [25], an ensemble of randomly constructed decision trees. In particular, we tested random forest with a variable number of random trees N, $N = 10$, $N = 25$, $N = 50$, $N = 75$ and $N = 100$.
- **Testing the model:** To test the approach, we measured *accuracy*, i.e., the total number of hits of the classifiers divided by the number of instances in the whole dataset:

$$Accuracy\ (Acc.) = \frac{TP + TN}{TP + FP + FN + TN} \qquad (11)$$

Besides, we measured the *Area Under the ROC Curve* (AUC), which establishes the relation between false negatives and false positives [26]. The ROC curve is obtained by plotting the TPR against the FPR. All these measures refer to the test instances.

TABLE II
RESULTS OF THE CATEGORIZATION USING BCLP AND SIMPLE FEATURE SETS BY THEMSELVES.

Classifier	BCLP		Simple	
	Acc.(%)	AUC	Acc.(%)	AUC
BN: K2	80.13	0.8787	91.95	0.9171
BN: TAN	93.33	0.8881	95.39	0.9185
Naïve Bayes	61.31	0.8216	79.94	0.9089
SVM: Poly. ker.	60.38	0.8441	90.13	0.9576
SVM: Norm. Poly. ker.	69.49	0.8650	92.62	0.9612
SVM: Pearson VII	89.33	0.8975	96.47	0.9749
SVM: RBF	60.15	0.8351	87.98	0.9528
KNN K=1	89.67	0.7674	92.62	0.5983
KNN K=2	91.28	0.8296	92.96	0.6342
KNN K=3	88.56	0.8558	93.05	0.6741
KNN K=4	89.43	0.8647	92.92	0.7088
KNN K=5	87.45	0.8715	92.95	0.7381
DT: J48	87.89	0.7735	93.89	0.8477
DT: Rand. For. N=10	91.41	0.9020	96.22	0.9626
DT: Rand. For. N=25	91.43	0.9159	96.46	0.9706
DT: Rand. For. N=50	91.75	0.9209	96.55	0.9743
DT: Rand. For. N=75	91.73	0.9224	96.60	0.9755
DT: Rand. For. N=100	91.78	0.9238	96.64	0.9763

Table II shows the results of the categorization using the different feature sets in order to compare which casting representation performs best and which classifier obtains the best results. In particular, Random Forest classifiers obtained the overall best results regardless the representation used, with accuracy and AUCs higher than 0.8 in both cases: (i) BCLP and (ii) Simple. At the contrary, although their results in accuracy are high, KNNs are the worst overall classifiers because they only classify correctly the correct castings and due to the unbalanced data the accuracy, which the percent of castings correctly classifier, is high, while the AUC shows a more realistic performance of the classifier.

TABLE III
RESULTS OF THE CATEGORIZATION USING THE COMBINATION OF SIMPLE FEATURE SET WITH BCLP.

Classifier	Simple+BCLP	
	Acc.(%)	AUC
BN: K2	92.30	0.8950
BN: TAN	94.13	0.8387
Naïve Bayes	78.45	0.8955
SVM: Polynomial kernel	91.47	0.9589
SVM: Normalized Polynomial kernel	92.79	0.9601
SVM: Pearson VII	95.88	0.9674
SVM: RBF	90.20	0.9553
KNN K=1	92.57	0.6038
KNN K=2	92.91	0.6450
KNN K=3	92.97	0.6804
KNN K=4	92.97	0.7089
KNN K=5	93.00	0.7323
DT: J48	93.84	0.8396
DT: Random Forest N=10	96.25	0.9622
DT: Random Forest N=25	96.46	0.9699
DT: Random Forest N=50	96.59	0.9732
DT: Random Forest N=75	96.59	0.9744
DT: Random Forest N=100	96.61	0.9751

Table III shows the results of the categorization using the combination of Simple feature set with BCLP. Again, Random Forest algorithms obtained the best results, with values higher than 96.25 and 0.96, in accuracy and AUC terms, in both cases: (i) BCLP and (ii) Simple. In the other hand, KNNs are the worst algorithms obtaining AUC values less than 0.7323.

VI. DISCUSSION AND CONCLUSIONS

In this paper, we proposed the application of Best Crossing Line Profile concept to our machine vision system. This technique, shows good results working with X-ray images. Concretely, it provides good information to distinguish between edge structures and defects in this kind of images.

The experimental results showed that, the features extracted from Best Crossing Line profile, provides lower results than simple features extracted from the initial images, but the combination of all, increases the accuracy and AUC results of the proposed classifiers.

Future work is oriented in 2 main ways. First, we are going to evaluate the performance of our system, using Best Crossing Line Profile features as a non supervised technique to distinguish the regular structures of the surface and defects. Second we will extract more features from the different representations of the surface information in order to get a complete information about the defects.

REFERENCES

[1] A. Mital, M. Govindaraju, and B. Subramani, "A comparison between manual and hybrid methods in parts inspection," *Integrated Manufacturing Systems*, vol. 9, no. 6, pp. 344–349, 1998.

[2] P. Kopardekar, A. Mital, and S. Anand, "Manual, hybrid and automated inspection literature and current research," *Integrated Manufacturing Systems*, vol. 4, no. 1, pp. 18–29, 1993.

[3] O. Silvén, M. Niskanen, and H. Kauppinen, "Wood inspectionwith non-supervisedclustering," *Machine Vision and Applications*, vol. 13, no. 5, pp. 275–285, 2003.

[4] V. Murino, M. Bicego, and I. Rossi, "Statistical classification of raw textile defects," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 4. IEEE, 2004, pp. 311–314.

[5] F. Pernkopf, "Detection of surface defects on raw steel blocks using bayesian network classifiers," *Pattern Analysis & Applications*, vol. 7, no. 3, pp. 333–342, 2004.

[6] Y. Frayman, H. Zheng, and S. Nahavandi, "Machine vision system for automatic inspection of surface defects in aluminum die casting," *Journal of advanced computational intelligence*, vol. 10, no. 3, pp. 281–286, 2011.

[7] D. Vom Stein, "Automatic visual 3-D inspection of castings," *Foundry Trade Journal*, vol. 180, no. 3641, pp. 24–27, 2007.

[8] R. Gonzalez and E. Richard, "Woods, digital image processing," *ed: Prentice Hall Press, ISBN 0-201-18075-8*, 2002.

[9] D. Mery, "Crossing line profile: a new approach to detecting defects in aluminium die casting," *Image Analysis*, pp. 245–256, 2003.

[10] E. Fix and J. Hodges Jr, "Discriminatory analysis-nonparametric discrimination: Small sample performance," DTIC Document, Tech. Rep., 1952.

[11] J. Iivarinen, J. Rauhamaa, and A. Visa, "Unsupervised segmentation of surface defects," in *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, vol. 4. IEEE, 1996, pp. 356–360.

[12] A. Bodnarova, J. Williams, M. Bennamoun, and K. Kubik, "Optimal textural features for flaw detection in textile materials," in *TENCON'97. IEEE Region 10 Annual Conference. Speech and Image Technologies for Computing and Telecommunications., Proceedings of IEEE*, vol. 1. IEEE, 1997, pp. 307–310.

[13] A. Monadjemi, "Towards efficient texture classification and abnormality detection," Ph.D. dissertation, University of Bristol, 2004.

[14] R. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 3, no. 6, pp. 610–621, 1973.

[15] C. Bishop *et al.*, *Pattern recognition and machine learning*. springer New York, 2006, vol. 4, no. 4.

[16] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[17] S. Garner, "Weka: The Waikato environment for knowledge analysis," in *Proceedings of the 1995 New Zealand Computer Science Research Students Conference*, 1995, pp. 57–64.

[18] G. Cooper and E. Herskovits, "A bayesian method for constructing bayesian belief networks from databases," in *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 1991, pp. 86–94.

[19] D. Geiger, M. Goldszmidt, G. Provan, P. Langley, and P. Smyth, "Bayesian network classifiers," in *Machine Learning*, 1997, pp. 131–163.

[20] S. Amari and S. Wu, "Improving support vector machine classifiers by modifying kernel functions," *Neural Networks*, vol. 12, no. 6, pp. 783–789, 1999.

[21] S. Maji, A. Berg, and J. Malik, "Classification using intersection kernel support vector machines is efficient," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2008, pp. 1–8.

[22] B. Üstün, W. Melssen, and L. Buydens, "Facilitating the application of support vector regression by using a universal pearson vii function based kernel," *Chemometrics and Intelligent Laboratory Systems*, vol. 81, no. 1, pp. 29–40, 2006.

[23] B. Cho, H. Yu, J. Lee, Y. Chee, I. Kim, and S. Kim, "Nonlinear support vector machine visualization for risk factor analysis using nomograms and localized radial basis function kernels," *IEEE Transactions on Information Technology in Biomedicine*, vol. 12, no. 2, pp. 247–256, 2008.

[24] J. Quinlan, *C4. 5: programs for machine learning*. Morgan kaufmann, 1993, vol. 1.

[25] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[26] Y. Singh, A. Kaur, and R. Malhotra, "Comparative analysis of regression and machine learning methods for predicting fault proneness models," *International Journal of Computer Applications in Technology*, vol. 35, no. 2, pp. 183–193, 2009.