Collective Classification for the Detection of Surface Defects in Automotive Castings

Iker Pastor-López, Igor Santos, Jorge de-la-Peña-Sordo,
Mikel Salazar, Aitor Santamaría-Ibirika, and Pablo G. Bringas
S^{3}Lab, DeustoTech - Computing, University of Deusto, Bilbao, Spain
Email: {iker.pastor, isantos, jorge.delapenya, mikel.salazar, a.santamaria, 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. For this reason, these castings are subject to very strict safety controls to ensure their final quality. 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 collective classification techniques to determine whether the regions are defective or not. We show that these classifiers obtain high precision rates whilst decreasing the effort of labelling.**

I. Introduction

One of the most important field in the industry sector is the foundry process, which consists in melting different types of material and pouring them into a mould where they solidify into the desired shape. Then, the produced castings are employed in other sectors such as automotive, aeronautic, 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.

There are a large number of defects that may appear in the manufactured castings. In this paper we focus on surface defects that occur in the external sides of the casting. Specifically, we deal with: (i) inclusions, little perforations caused by an excess of sand in the mould; (ii) cold laps, produced when part of the melted material is cooled down before the melting is completed; and (iii) misruns, which appear when not enough material is poured into the mould.

Nowadays, the inspection of the quality of the castings is performed by human operators [1]. Albeit people can perform some tasks better than machines, they are slower and get easily exhausted. Besides, qualified operators are hard to find and to maintain in the industry since they require capabilities and learning skills that usually take long time to learn. There are also cases in whose boredom affected the process. In some applications, the inspection is critical and dangerous and computer vision can replace manual inspection more efficiently and safely [2].

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.

In our previous work, we proposed a new approach for surface defect detection and categorisation using machine-learning classifiers [7]. Nevertheless, these supervised classifiers require a high number of labelled instances for each class. In a real-world problem such a quality inspection in foundries, it is quite cumbersome to collect this number of labelled data.

Semi-supervised learning is a type of machine-learning that is useful when a limited amount of labelled data for each class is available [8]. Concretely, this approach uses the relational structure of the combined labelled and unlabelled data-sets to increase the classification accuracy [9]. With these relational approaches, the predicted label of an example will often be influenced by the labels of related samples. Collective classifiers and semi-supervised techniques have been used successfully in fields like text classification [9], malware detection [10], [11] or spam filtering [12]. The main advantage of collective classification is that the predicted labels of a test sample should also be influenced by the predictions made for related test samples.

In light of this background, we propose a new approach capable of detecting 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 modelling 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 collective classifiers to detect 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 collective classifiers based approach to determine faulty regions on the foundry castings and (iii) an empirical validation using actual foundry castings of our proposed approach.

II. Data Acquisition and Segmentation through Machine Vision

In order to retrieve and process the casting model data, we develop a simple computer-vision system that is composed of a laser camera with 3D technology, a computer with high data processing capabilities and a robotic arm, already employed in our previous work [7].

978-1-4673-6322-8/13/$31.00 © 2013 IEEE

In our case, the following components compose our system:

1) **Image device:** We obtain the three-dimensional data through a laser-based triangulation camera. By taking advantage of the high-power laser (3-B class), our system is able to scan the casting even though their dark and uneven surface.
2) **Processing device:** We utilise a high-speed workstation. In particular, we have used a workstation with a XENON E5506 processor working with 6GB of RAM memory and a QUADRO FX1800 graphic processing unit. This component controls the camera and the robotic arm. It also processes the information retrieved by the image capturing device and transforms it to segments.
3) **Robotic arm:** The function of the robot is to automate the topology information gathering phase of the system, making every necessary movement to successfully acquire the data and attaching the camera.

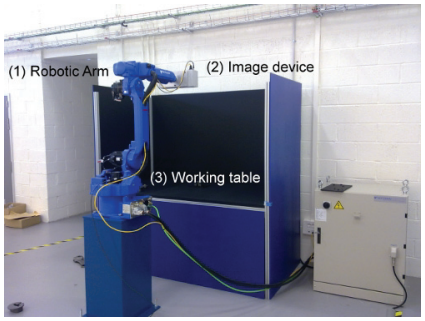


Fig. 1. The architecture of the machine vision system. (1) is the robotic arm of the system, (2) is the image device of the system and (3) is the working table where the castings are put for analysis.

The casting is put on a working table using a manually adjusted foundry mould. The mould has been built using a material similar to common silicone, which is easily malleable. In the case we decide to change the casting type, we would only need to change this mould. In this way, we ensure that the vision system allows us to analyse every type of casting in the same position.

Using this architecture, we gather the topological information of the castings. Initially, we put the casting on the mould and we start the surface scan. The robotic arm makes a lineal movement, retrieving a set of profiles based on the generated triangulation of the laser and the optical sensor. In other words, a foundry casting $\mathcal{C}$ is composed of profiles $\mathcal{P}$ such as $\mathcal{C} = \{\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_{n-1}, \mathcal{P}_n\}$. Each profile is retrieved for a thickness of 0.2mm. These profiles are vectors $\vec{p}$ composed of the heights of each point $p_{x,y}$. Joining these profiles, we represent the casting $\mathcal{C}$ as a height matrix $\mathcal{H}$

$$\mathcal{H} = \begin{pmatrix} h_{1,1} & h_{1,2} & ... & h_{1,m-1} & h_{1,m} \\ h_{2,1} & h_{2,2} & ... & h_{2,m-1} & h_{2,m} \\ & & ... & & \\ h_{\ell-1,1} & h_{\ell-1,2} & ... & h_{\ell-1,m-1} & h_{\ell-1,m} \\ h_{\ell,1} & h_{\ell,2} & ... & h_{\ell,m-1} & h_{\ell,m} \end{pmatrix} \quad (1)$$

where each $h_{x,y}$ represent the height of the point in the space (x, y). Therefore, the number of profiles of each casting depends on its size.

Once the system has computed the matrix $\mathcal{H}$, we have to remove any possible remaining noise, as well as the data unrelated to the casting surface, like the working table. To this end, we establish a height threshold, established empirically.

We have also taken into consideration other representations of the data using the height matrix: (i) *Grey-scale Height Map* [13], a well-known representation converts each height value to a range between 0 and 255, showing the different scales of grey; (ii) *Colour Height Map*, a representation similar to the Grey-scale Height Map but with higher detail, since full colour information is used; specifically, the RGB (i.e., Red, Green and Blue) components and the colour scale Jet Colour Map as defined in Matlab [14]; (iii) *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.

Thenceforth, the system starts with the segmentation process. Segmentation is the process of selecting the regions of the surface of the casting that may have defects. To this end, we made two different labelling tasks. The first one classifies each casting as good or potentially defective. This classification is necessary for the construction of the models used in the segmentation process. The second one is for evaluating the accuracy of the segmentation and labels each segment in 'correct' or 'defective' (the defect may be misrun, cold lap or inclusion).

The segmentation process is accomplished by the following steps:

1) The system converts the normal map to grey-scale — although it is not an image, we represent it 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 [15] is applied.
3) The system applies a difference filter between the normal map to examine and each of the correct models.
4) The system applies an intersection filter between the differences computed in the previous step.
5) The system binarises the results.
6) The system uses an algorithm to extract the potentially faulty areas or segments, removing the ones which are excessively small (i.e., regions smaller than 3x3 pixels).

Once each area is extracted, we compute several features using the different representations:

- **Features of the segmented image:** The segmented image is the result of the segmentation process applied to the normal map. We use: (i) the width, height and perimeter of the area; (ii) the euclidean distance of the center of gravity of the area to origin of coordinate axes; and

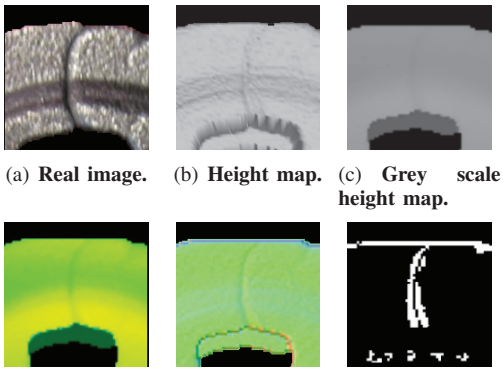
(iii) the fullness, which is computed as $Area/(Width * Height)$.

- **Features of the integral image of segmented binary image:** These features are obtained from the conversion to the integral image of the segmented version of the image. An integral image is defined as the image in which the intensity at a pixel position is equal to the sum of the intensities of all the pixels above and to the left of that position in the original image [16]. We use: (i) mean value of pixels in the integral image and (ii) the result of addition of the pixels values in the integral image.
- **Features of the grey scale height map:** They are extracted from the computed segments in the original grey-scale height map. We use: (i) max, min, mean, median, standard deviation and entropy of the grey histogram values; and (ii) max, min, mean, median, standard deviation and entropy of the grey histogram values without black (zero value) pixels.
- **Features of the colour height map:** These features are extracted from the computed segments in the original colour height map. We use: (i) max, min, mean, median, standard deviation and entropy of the red histogram values; (ii) max, min, mean, median, standard deviation and entropy of the red histogram values without black (zero value) pixels; (iii) max, min, mean, median, standard deviation and entropy of the green histogram values; (iv) max, min, mean, median, standard deviation and entropy of the green histogram values without black (zero value) pixels; (v) max, min, mean, median, standard deviation and entropy of the blue histogram values; and (vi) max, min, mean, median, standard deviation and entropy of the blue histogram values without black (zero value) pixels.
- **Features of the normal map:** These features are extracted from the computed segments in the original normal map. We use: (i) max, min, mean, median, standard deviation and entropy of the x component histogram values; (ii) max, min, mean, median, standard deviation and entropy of the x component histogram values without black (zero value) pixels; (iii) max, min, mean, median, standard deviation and entropy of the y component histogram value; (iv) max, min, mean, median, standard deviation and entropy of the y component histogram values without black (zero value) pixels; (v) max, min, mean, median, standard deviation and entropy of the z component histogram values; and (vi) max, min, mean, median, standard deviation and entropy of the z component histogram values without black (zero value) pixels.

By extracting these features from each segment, we can train collective machine-learning algorithms in order to determine weather a potentially faulty segment is defective or not.

III. Collective Classification

Collective classification is a combinatorial optimisation problem, in which we are given a set of castings, or nodes, $\mathcal{E} = \{e_1, ..., e_n\}$ and a neighbourhood function N, where $N_i \subseteq \mathcal{E} \setminus \{\mathcal{E}_i\}$, which describes the underlying network structure [17]. Being $\mathcal{E}$ a random collection of castings, it is divided into two sets $\mathcal{X}$ and $\mathcal{Y}$, where $\mathcal{X}$ corresponds to the castings for which we know the correct values and $\mathcal{Y}$ are the castings whose values need to be determined. Therefore, the task is to label the nodes $\mathcal{Y}_i \in \mathcal{Y}$ with one of a small number of labels, $\mathcal{L} = \{l_1, ..., l_q\}$.



(a) **Real image.** (b) **Height map.** (c) **Grey scale height map.**

(d) **Colour scale height map.** (e) **Normal map.** (f) **Segmented image.**

Fig. 2. Visual representations of a cold lap region.

We use the *Waikato Environment for Knowledge Analysis* (WEKA) [18] and its Semi-Supervised Learning and Collective Classification plugin[1]. In the remainder of this section we review the collective algorithms used in the empirical evaluation.

- **CollectiveIBK.** Internally, this model uses WEKA's classic IBK algorithm, an implementation of the *K-Nearest Neighbour* (KNN), to determine the best k instances on the training set and builds then, for all instances from the test set, a neighbourhood consisting of k instances from the training pool and test set (either a naïve search over the complete set of instances or a k-dimensional tree is used to determine neighbours). All neighbours in such a neighbourhood are sorted according to their distance to the test instance they belong to. The neighbourhoods are sorted according to their 'rank', where 'rank' means the different occurrences of the two classes in the neighbourhood.

  For every unlabelled test instance with the highest rank value, the class label is determined by majority vote or, in case of a tie, by the first class. This is performed until no further test instances remain unlabelled. The classification terminates by returning the class label of the instance that is about to be classified.
- **CollectiveForest.** It uses WEKA's implementation of RandomTree as base classifier to divide the test set into folds containing the same number of elements. The first iteration trains the model using the original training set and generates the distribution for all the instances in the test set. The best instances are then added to the original training set (being the number of instances chosen the same as in a fold).

[1]Available at: http://www.scms.waikato.ac.nz/~fracpete/projects/collectiveclassification

The next iterations train the model with the new training set and generate then the distributions for the remaining instances in the test set.

- **CollectiveWoods & CollectiveTree.** CollectiveWoods works like CollectiveForest using CollectiveTree algorithm instead of RandomTree.
  Collective tree is similar to WEKA's original RandomTree classifier. It splits the attribute at a position that divides the current subset of instances (training and test instances) into two halves. The process finishes if one of the following conditions is met: (i) only training instances are covered (the labels for these instances are already known); (ii) only test instances in the leaf, case in which distribution from the parent node is taken, and (iii) only training instances of one class, case in which all test instances are considered to have this class.
  To calculate the class distribution of a complete set or a subset, the weights are summed up according to the weights in the training set, and then normalised. The nominal attribute distribution corresponds to the normalised sum of weights for each distinct value and, for the numeric attribute, distribution of the binary split based on median is calculated and then the weights are summed up for the two bins and finally normalised.
- **RandomWoods** It works like WEKA's classic RandomForest but using CollectiveBagging (classic Bagging, a machine learning ensemble meta-algorithm to improve stability and classification accuracy, extended to make it available to collective classifiers) in combination with CollectiveTree. RandomForest, in contrast, uses Bagging and RandomTree algorithms.

IV. Empirical Validation

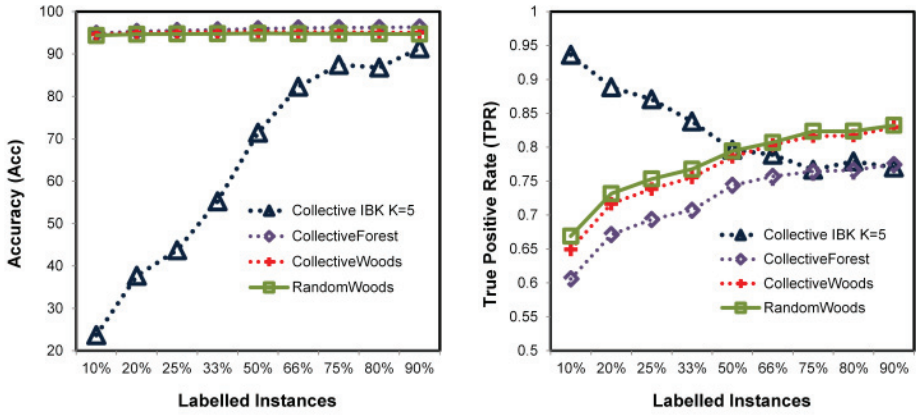
In order to evaluate our casting defect detector, we collected a dataset from a foundry, which is specialised in safety and precisions components for the automotive industry (principally, in disk-brake support with a production over 45,000 tons per year). Three different types of defect (i.e., inclusion, cold lap and misrun) were present in the faulty castings.

To construct the dataset, we analysed 645 foundry castings with the segmentation machine-vision system described in Section II in order to retrieve the different segments and their features. In particular, we used 176 correct castings to construct the model and the remainder for testing. By means of this analysis, we constructed a dataset of 6150 segments to train collective models and determine when a segment is defective. Besides, we added a second category to identify the noise that our machine vision system retrieves called 'Correct', which represents the segments gathered 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.

The acceptance/rejection criterion of the studied models resembles the one applied by the final requirements of the customer. Pieces flawed with defects must be rejected due to the very restrictive quality standards (which is a requirement of the automotive industry). We labelled each possible segment within the castings with its defects.

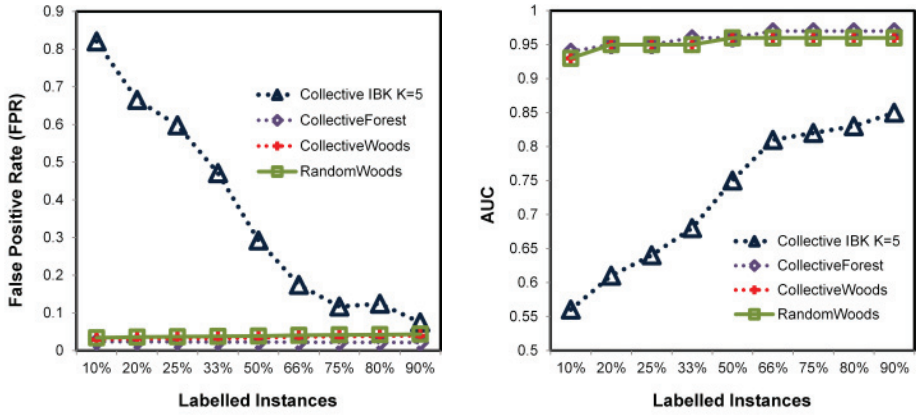
Next, we evaluate the precision of the collective machine-learning methods to categorise the segments. To this extent, by means of the dataset, we conducted the following methodology to evaluate the proposed method:

- **Cross validation:** This method is generally applied in machine-learning evaluation [19]. 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 labelled instances from 10% to 90% to measure the effect of the number of previously labelled 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) [20], 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 using different learning algorithms depending on the specific model. As aforementioned, we used the collective classification implementations provided by the *Semi-Supervised Learning and Collective Classification* package for machine-learning tool WEKA [18] In particular, we used the following models:
  - *Collective IBK:* We performed experiments with $k = 5$.
  - *Collective Forest:* We performed experiments with 100 trees, the default value of classifier.
  - *Collective Woods:* We performed experiments with 100 trees, the default value of classifier.
  - *Random Woods:* We performed experiments with 100 trees, the default value of classifier.
- **Testing the model:** To test the approach, we measured the *True Positive Rate* (TPR), i.e., the number of the faulty segments correctly detected divided by the total number of faulty segments: $TPR = TP/(TP + FN)$ where TP is the number of faulty instances correctly classified (true positives) and FN is the number of faulty instances misclassified as correct segments (false negatives).
  We also measured the *False Positive Rate* (FPR), i.e., the number of correct segments misclassified as faulty divided by the total number of correct segments: $FPR = FP/(FP + TN)$ where FP is the number of correct instances incorrectly detected as faulty and TN is the number of correct segments correctly classified.
  Furthermore, we measured *accuracy*, i.e., the total number of hits of the classifiers divided by the number of instances in the whole dataset: $Accuracy(\%) = (TP + TN)/(TP + FP + TP + TN)$
  Besides, we measured the *Area Under the ROC Curve*



(a) **Accuracy results.** The classifiers were sensitive to the number of instances used in the training step. With the exception of Collective IBK, the rest of the classifiers obtained accuracies higher than 90% using only a 10% of labelled instances.

(b) **TPR results.** In general, the higher the amount of labelled instances, the higher the TPR. The worst classifier is Collective forest, although more of the 30% of labelled instances Collective Forest obtained TPR values higher than 75%.



(c) **FPR results.** Collective IBK was the worst classifier with a minimun FPR of 0.0749% using the 90% of the dataset. The remainder of the classifiers obtained a FPR of less than 0.04% with only a 10% of training size.

(d) **AUC results.** The AUC axis (Y axis) has been scaled from 50% to 100% in order to appreciate better the evolution of the classifiers. As it happened with accuracy, Collective IBK was the worst classifier.

Fig. 3. Results of our collective-classification-based for surface defect detection. Collective Forest was the overall classifier with the highest accuracy, TPR and AUC.

(AUC), which establishes the relation between false negatives and false positives [21]. The ROC curve is obtained by plotting the TPR against the FPR. All these measures refer to the test instances.

Figure 3 shows the obtained results in terms of accuracy, TPR, FPR and AUC. Our results outline that, as is usually the case, the higher the number of labelled instances in the dataset the better results achieved. However, by using only the 10% of the available data, with the exception of Collective IBK and Collective Forest, the collective classifiers were able to achieve TPRs higher than 75% and FPRs lower than 4%. In particular, Random Woods trained with the 50% of the data obtained 94.88% of accuracy, 79.44% of TPR, 3.86% of FPR and 95.77% of AUC. Figure 3(a) shows the accuracy results of our proposed method. All the tested classifiers, with the exception of Collective IBK, achieved accuracy results higher than 90%. In particular, Collective Forest was the best, achieving an accuracy of 94.84% using only a 10% of the instances for training and 95,97% using a 50% of labelled instances. Figure 3(b) shows the obtained results in terms of correctly classified faulty segments. In this way, Random Woods was also the best detecting the 79.44% of the faulty castings with only a 50% of the dataset labelled. Figure 3(c) shows the FPR results. Every classifier obtained results lower than 5%, except Collective IBK. In particular, the lowest FPR achieved was of 2.36%, achieved by Collective Forest with the 10% of dataset. Finally, regarding AUC, shown in Figure 3(d), Collective Forest was again the best, with results higher than 94% for every configuration.

Table I shows the results for the classification task using supervised learning techniques. In this way, we can compare these results with the ones achieved using collective techniques. Regarding supervised classification Random Forest with N=100 was the best classifier, accomplishing a 96.15%

TABLE I
RESULTS OF THE DEFECT DETECTION IN TERMS OF ACCURACY, TRUE POSITIVE RATE, FALSE POSITIVE RATE AND AUC. THE BEST RESULTS WERE OBTAINED BY THE RANDOM FOREST TRAINED WITH MORE THAN 50 TREES.

Model	Accuracy (%)	TPR	FPR	AUC
Bayes K2	93.01 ± 1.00	0.5982 ± 0.07	0.0423 ± 0.01	0.8845 ± 0.03
Bayes TAN	92.54 ± 0.27	0.0291 ± 0.03	0.0015 ± 0.00	0.8672 ± 0.03
Naïve Bayes	79.34 ± 1.59	0.8698 ± 0.05	0.2128 ± 0.02	0.8811 ± 0.03
SVM: Polynomial Kernel	90.22 ± 1.07	0.8767 ± 0.05	0.0957 ± 0.01	0.9533 ± 0.01
SVM: Normalised Polynomial Kernel	91.61 ± 0.99	0.8659 ± 0.05	0.0798 ± 0.01	0.9563 ± 0.01
SVM: Pearson VII Kernel	95.95 ± 0.76	0.7227 ± 0.06	0.0212 ± 0.01	0.9665 ± 0.01
SVM: Radial Basis Function Kernel	88.78 ± 1.29	0.8918 ± 0.04	0.1125 ± 0.01	0.9511 ± 0.02
KNN K = 1	94.48 ± 0.84	0.7626 ± 0.06	0.0403 ± 0.01	0.8611 ± 0.03
KNN K = 2	95.13 ± 0.76	0.6951 ± 0.07	0.0278 ± 0.01	0.9029 ± 0.03
KNN K = 3	94.02 ± 0.91	0.8107 ± 0.05	0.0492 ± 0.01	0.9199 ± 0.03
KNN K = 4	94.44 ± 0.85	0.7766 ± 0.06	0.0419 ± 0.01	0.9322 ± 0.02
KNN K = 5	93.30 ± 0.86	0.8286 ± 0.05	0.0585 ± 0.01	0.9322 ± 0.02
J48	93.21 ± 0.97	0.7143 ± 0.06	0.0501 ± 0.01	0.8240 ± 0.05
Random Forest N = 10	95.83 ± 0.69	0.7392 ± 0.07	0.0238 ± 0.01	0.9524 ± 0.02
Random Forest N = 25	95.86 ± 0.74	0.7804 ± 0.06	0.0269 ± 0.01	0.9628 ± 0.02
Random Forest N = 50	96.13 ± 0.72	0.7804 ± 0.06	0.0239 ± 0.01	0.9664 ± 0.01
Random Forest N = 75	96.09 ± 0.73	0.7851 ± 0.06	0.0247 ± 0.01	0.9670 ± 0.01
Random Forest N = 100	96.15 ± 0.73	0.7824 ± 0.06	0.0238 ± 0.01	0.9676 ± 0.01

of accuracy, 78.24% of TPR, 2.38% of FPR and 96.76% of AUC. If we compare these results with the ones obtained by Random Woods with a 50% of the labelled dataset: 94.88% of accuracy, 79.34% of TPR, 3.86% of FPR and 95.77% of AUC, we can notice that by only using the 50% of the information that the supervised approach employs we can guarantee that the results will be nearly as high as the ones obtained by them.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a novel collective-classification-based approach for surface defect detection in iron castings. This method requires less labelling effort than presented previous work [7] based on supervised learning approach.

Future work will be focused on three main directions. First, we will utilise different features and segmentation methods for training these kinds of models. Second, we will extend our study of collective learning by applying more algorithms to this issue. Finally, we are going to focus on different defects in foundry production in order to generate a general fault detector.

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] I. Pastor-López, I. Santos, A. Santamaría-Ibirika, M. Salazar, J. de-la-Peña Sordo, and P. Bringas, "Machine-learning-based surface defect detection and categorisation in high-precision foundry," in *Proceedings of 7th IEEE Conference in Industrial Electronics and Applications (ICIEA)*, 2012.

[8] O. Chapelle, B. Schölkopf, A. Zien *et al.*, *Semi-supervised learning*. MIT press Cambridge, MA:, 2006, vol. 2.

[9] J. Neville and D. Jensen, "Collective classification with relational dependency networks," in *Proceedings of the Second International Workshop on Multi-Relational Data Mining*, 2003, pp. 77–91.

[10] I. Santos, C. Laorden, and P. Bringas, "Collective classification for unknown malware detection," in *Proceedings of the 6th International Conference on Security and Cryptography (SECRYPT)*, 2011, pp. 251–256.

[11] I. Santos, J. Nieves, and P. G. Bringas, "Semi-supervised learning for unknown malware detection." in *Proceedings of the 4^{th} International Symposium on Distributed Computing and Artificial Intelligence (DCAI). 9th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS)*, 2011, pp. 415–422.

[12] C. Laorden, B. Sanz, I. Santos, P. Galán-García, and P. G. Bringas, "Collective classification for spam filtering," in *Computational Intelligence in Security for Information Systems*, ser. Lecture Notes in Computer Science, vol. 6694. Springer, 2011, pp. 1–8.

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

[14] R. Gonzalez, R. Woods, and S. Eddins, *Digital image processing using MATLAB*. Pearson Education India, 2004.

[15] R. Gonzalez and R. Woods, "Digital image processing. 1992," *Reading, Mass.: Addison-Wesley*, vol. 16, no. 716, p. 8.

[16] P. Viola and M. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

[17] G. Namata, P. Sen, M. Bilgic, and L. Getoor, "Collective classification for text classification," *Text Mining*, pp. 51–69, 2009.

[18] 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.

[19] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[20] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, no. 3, pp. 321–357, 2002.

[21] 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.