Enhanced Image Segmentation using Quality Threshold Clustering for Surface Defect Categorisation in High Precision Automotive Castings

Iker Pastor-López, Igor Santos, Jorge de-la-Peña-Sordo, Iván García-Ferreira, Asier G. Zabala, and Pablo G. Bringas

S^{3}Lab, DeustoTech - Computing, University of Deusto, Bilbao, Spain
{iker.pastor, isantos, jorge.delapenya, ivan.garcia.ferreira, asier.gonzalez, pablo.garcia.bringas}@deusto.es

Abstract. Foundry is an important industry that supplies key products to other important sectors of the society. In order to assure the quality of the final product, the castings are subject to strict safety controls. One of the most important test in these controls is surface quality inspection. In particular, our work focuses on three of the most typical surface defects in iron foundries: inclusions, cold laps and misruns. In order to automatise this process, we introduce the QT Clustering approach to increase the perfomance of a segmentation method. Finally, we categorise resulting areas using machine-learning algorithms. We show that with this addition our segmentation method increases its coverage.

Keywords: Computer Vision, Machine-Learning, Defect Categorisation, Foundry

1 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. Later, the resulting castings are used in other industries like aeronautic, automotive, weaponry or naval, where they are critical and any defect may be critical. For this reason, the manufactured castings must success very strict safety controls to ensure their quality.

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 are performed by human operators [1, 2]. Although, people can perform some tasks better than machines, they are slower and can get easily exhausted. In addition, qualified

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.

2 Proposed Machine Vision System

For the casting surface information retrieval, a simple computer-vision system was developed, composed by [3]: (i) image device, (ii) processing device and (iii) robotic arm.

1. **Image device:** We obtain the three-dimensional data through a laser-based triangulation camera. By taking advantage of the high-power (3-B class) laser, we are able to scan the casting even though their surface tends to be dark.
2. **Processing device:** We utilise a high-speed workstation. In particular, we use 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. Besides, it processes the information retrieved by the image capturing device and transforms it into segments.
3. **Robotic arm:** The function of the robot is to automate the gathering phase of the system, making every necessary move to successfully acquire the data. There are two working options [4]: (i) to use the arm in order to handle the tested castings, leaving the image device in a fixed position or (ii) to attach the camera to the robotic arm. We selected the second one due to the diversity of the castings.

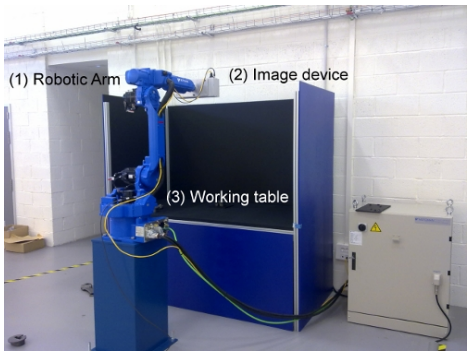

Fig. 1: The architecture of the machine vision system. (1) is the robotic arm, (2) is the image device and (3) is the working table where the castings are put for analysis.

The casting is positioned on a working table using a manually adjusted foundry mould. The mould is built with a material similar to common silicone,

which is easily malleable. In the case that we decide to change the casting type, we will only have to change the mould. In this way, we ensure that the vision system allows us to analyse every type of casting in the same position.

With this system, we capture the information of the casting surface. The process starts by putting the casting manually on a working table. Then, we use a mould that is built with a material similar to common silicone. In this way, we ensure that the vision system allows us to analyse every type of castings in the same position.

When the casting is on the working table, the robotic arm makes a linear 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 with a thickness of 0.2mm. These profiles are vectors $\boldsymbol{p}$ 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 the possible existing noise, as well as the data unrelated to the casting surface. To this end, we establish a height threshold empirically.

Finally, we generate the following representations of the information, besides from the heigh matrix:(i) normals matrix and (ii) normals map coded in RGB.

- **Normals Matrix.** This representation is generated by means of the height matrix, but shows the direction of the normal vector of the surface for each point in the matrix. Each vector for each point have three components (x, y, z).
- **Normals Map coded in RGB.** This image represents the information of the Normals Matrix corresponding red component to the x value, green component to the y value and blue component to the z value.

3 Enhanced Segmentation Method

Image segmentation consists on the subdivision of an image into disjointed regions [5]. Usually, these regions represent areas of the original image that contain at least one irregularity (defect or regular structure). In [6], we presented a model based approach for image segmentation. This method uses correct castings to

compare with the normals map coded in RGB of the potentially defective surfaces. Then, we employed 176 correct castings to build the model, and we confirmed that if the number of correct castings increases, the performance of the segmentation method decreases.

In this paper we optimise this segmentation method, using a combination of: (i) image filters (to emphasise the defective areas); and (ii) the use of QT clustering algorithm (to reduce the information of the good castings used like a models).

Quality threshold algorithm was proposed by Heyer et al. [7] to extract relevant information of big datasets. Specifically, we use the implementation of this algorithm proposed by Ugarte-Pedrero et al. [8]. The Figure 2 shows the pseudo-code of this implementation and how are centroid vectors generated.

```
input  : The original dataset V, the distance threshold for each cluster threshold, and the
         minimum number of vectors in each cluster minimumvectors
output: The reduced dataset R
// Calculate the distance from each vector (set of executable features) to the rest of
   vectors in the dataset.
foreach {v_i | v_i ∈ V} do
    foreach {v_j | v_j ∈ V} do
        // If a vector v_j's distance to v_i is lower than the specified threshold, then
           v_j is added to the potential cluster A_i, associated to the v_i vector
        if distance(v_i,v_j) ≥ threshold then
            A_i.add(v_j)
        end
    end
end
// In each loop, select the potential cluster with the highest number of vectors.
while ∃A_i ∈ A : |A_i| ≥ minimumvectors and ∀A_j ∈ A : |A_i| ≥ |A_j| and i ≠ j do
    // Add the centroid vector for the cluster to the result set R.add(centroid(Ai))
    R.add(centroid(A_i))
    // Discard potential clusters associated to vectors v_j ∈ A_i
    foreach {v_j | v_j ∈ A_i} do
        A.remove(A_j) V.remove(v_j)
    end
    // Remove vectors v_j ∈ A_i from the clusters A_k remaining in A
    foreach {A_k | A_k ∈ A} do
        foreach {v_j | v_j ∈ A_k and v_j ∈ A_i} do
            A_k.remove(v_j)
        end
    end
end
// Add the remaining vectors to the final reduced dataset
foreach {v_j | v_j ∈ V} do
    R.add(v_j)
end
```

Fig. 2: Pseudo-code of the implementation of QT Clustering based model reduction algorithm proposed by Ugarte-Pedrero et al. [8].

The input vectors are composed of the heigh matrix of the correct casting surfaces, concatenating each row in a sole row. In other words, if we have a height matrix $\mathcal{H}$, the input vector $\boldsymbol{v}$ that represents $\mathcal{H}$ is the following

$$\boldsymbol{v} = \{h_{1,1}, h_{1,2}, ..., h_{1,n-1}, h_{1,n}, h_{2,1}, h_{2,2}, ..., h_{m,n-1}, h_{m,n}\} \tag{2}$$

QT clustering algorithm requires to fix a threshold to determine the maximum distance between two vector of the same cluster. Specifically, we use Euclidean distance and we set different values for the threshold to optimise the performance of the segmentation method.

Next, the centroids are generated, replacing the original model castings and the segmentation process continues with the following steps:

1. The process starts converting to grey-scale the normals map coded in RGB of the casting and of the correct models. This step is necessary to remove any noise of the rugosity of the surface.
2. The Gaussian Blur [9] filter is applied.
3. The process continues applying the difference filter between the result image of the previous steps and each model image.
4. The system applies a intersection filter between the differences computed in the previous step.
5. The result image is binarized.
6. The process ends with an algorithm that extracts the areas potentially faulty, removing the ones which are excessively small.

For each extracted area, several features are computed. These features can be divided into the following categories:

- **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 [10]. 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 height matrix:** They are extracted from the computed segments in the original grey-scale height map. We use: (i) summation, mean, variance, standard deviation, standard error, min, max, range, median, entropy, skewness and kurtosis of the height matrix values; and (ii) summation, mean, variance, standard deviation, standard error, min, max, range, median, entropy, skewness and kurtosis of the height matrix without zero pixels values.
- **Features of the normals matrix:** These features are extracted from the computed segments in the original normals matrix. We use: (i) summation, mean, variance, standard deviation, standard error, min, max, range, median, entropy, skewness and kurtosis of the x component; (ii) summation, mean, variance, standard deviation, standard error, min, max, range, median, entropy, skewness and kurtosis of the x component without zero pixels

values; (iii) summation, mean, variance, standard deviation, standard error, min, max, range, median, entropy, skewness and kurtosis of the y component; (iv) summation, mean, variance, standard deviation, standard error, min, max, range, median, entropy, skewness and kurtosis of the y component without zero pixels values; (v) summation, mean, variance, standard deviation, standard error, min, max, range, median, entropy, skewness and kurtosis of the z component; and (vi) summation, mean, variance, standard deviation, standard error, min, max, range, median, entropy, skewness and kurtosis of the z component without zero pixels values.

4 Empirical validation

To evaluate our casting defect detector and categoriser, 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 639 foundry castings with the segmentation machine-vision system described in Section 2 in order to retrieve the different segments and their features. In particular, we used 236 correct castings as input for the clustering algorithm and the remainder for testing.

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.

First, we evaluate the coverage of our segmentation method using different values for QT Clustering threshold. To this end, we define the metric 'Coverage' as:

$$Coverage = \frac{S_{s \to s}}{S_{s \to s} + S_{c \to s}} \cdot 100 \tag{3}$$

where $S_{s \to s}$ is the number of segments retrieved by the segmentation system which are defects and $S_{c \to s}$ are the number of defects that our segmentation method does not gather.

The Table 1 shows the evolution of the coverage of the segmentation method using different values for the clustering threshold.

We can notice that the coverage increases with higher values of the threshold. Besides, when we use a threshold higher than 450, the segmented areas are too big and the method loses precision. For this reason, we compute the segmentation process using the 51 centroids vectors.

By means of this analysis, we constructed a dataset of 6,150 segments to train machine-learning 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

Table 1: Coverage results and generated centroids for different threshold values.

Threshold	*Number of centroids*	*Coverage*(%)
300	236	59.71
325	234	59.87
350	218	60.69
375	182	62.97
400	128	66.72
425	68	77.00
450	51	78.79

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.

Table 2: Number of samples for each category.

Category	*Number of samples*
Correct	33,216
Inclusion	553
Cold Lap	20
Misrun	60

Next, we evaluate the precision of the 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 [11]. 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 (90% of the total dataset) and testing (10% of the total dataset).
- **SMOTE:** The dataset was not balanced for the different classes. To address unbalanced data, we applied Synthetic Minority Over-sampling TEchnique (SMOTE) [12], 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. Particularly, we used the following models:
  - *Bayesian networks (BN):* With regards to Bayesian networks, we utilize different structural learning algorithms: K2 [13] and Tree Augmented Naïve (TAN) [14]. Moreover, we also performed experiments with a Naïve Bayes Classifier [11].

- *Support Vector Machines (SVM):* We performed experiments with a polynomial kernel [15], a normalized polynomial Kernel [16], a Pearson VII function-based universal kernel [17] and a radial basis function (RBF) based kernel [18].
- *K-nearest neighbour (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* [19] implementation of the *C4.5* algorithm [20]) and Random Forest [21], 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 evaluated the percent of correctly classified instances and the area under the ROC curve, which establishes the relation between false negatives and false positives [22].

Regarding the coverage results, our segmentation method is able to detect 78.79% of the surface defects. This coverage value is higher than we obtained without clustering. In particular, the coverage increases in 19.09 points.

Table 3: Results of the categorisation in terms of accuracy and AUC.

Model	**Accuracy(%)**	**AUC**
Bayes K2	97.21	0.8364
Bayes TAN	98.40	0.7229
Naïve Bayes	81.83	0.8938
SVM: Polynomial Kernel	93.22	0.9543
SVM: Normalised Polynomial Kernel	96.85	0.9611
SVM: Pearson VII Kernel	98.81	0.9516
SVM: Radial Basis Function Kernel	93.98	0.9578
KNN K = 1	98.03	0.5584
KNN K = 2	98.17	0.5860
KNN K = 3	98.11	0.6104
KNN K = 4	98.16	0.6277
KNN K = 5	98.09	0.6450
J48	97.58	0.7911
Random Forest N = 10	98.63	0.9497
Random Forest N = 25	98.62	0.9621
Random Forest N = 50	98.66	0.9680
Random Forest N = 75	98.67	0.9692
Random Forest N = 100	98.70	0.9689

If we focus in the precision of the categorisation of the segments, Table 3 shows the results of the categorisation phase. In particular, the best results were obtained by the Random Forest trained with more than 50 trees with an

accuracy of more than 98% and an AUC of 0.96. SVM trained with a Radial Basis Function kernel and trained with Polynomial Kernel obtained poor results, implying that a radial division of the space is not as feasible as others, because the rest of the SVMs behaved with accuracies higher 98% in the case of Pearson VII and near 97% in the case of the Normalised Polynomial kernel. Surprisingly, the lazy classifier KNN achieved high results, ranging from 98.03% to 98.17% of accuracy and from 0.55 to 0.64 or AUC. J48 was an average classifier that achieved an AUC of 0.79.

5 Conclusions and future work

In this paper, we proposed an improvement for a machine vision system. Concretely, we used Quality Threshold Clustering to reduce the data of the correct castings used in the segmentation methods. Also, with this enhancement we have increased the coverage of the method. Then we evaluated our new segmentation method using machine learning models to categorise the detected areas into correct, inclusion, cold lap or misrun. For this classification, we proposed new features, using different representations. The experimental results showed that, albeit our precision in categorisation is very high, the coverage of the segmentation method had increased.

Future work is oriented in 2 main ways. First, we are going to develop new segmentation methods in order to enhance the coverage results and the system performance. Second, we will evaluate different features and approaches in order to improve the categorisation process.

References

1. Mital, A., Govindaraju, M., Subramani, B.: A comparison between manual and hybrid methods in parts inspection. Integrated Manufacturing Systems **9**(6) (1998) 344–349
2. Watts, K.P.: The effect of visual search strategy and overlays on visual inspection of castings. Master's thesis, Iowa State University (2011)
3. Pernkopf, F., O'Leary, P.: Image acquisition techniques for automatic visual inspection of metallic surfaces. NDT & E International **36**(8) (2003) 609–617
4. vom Stein, D.: Automatic visual 3-d inspection of castings. Foundry Trade Journal **180**(3641) (2007) 24–27
5. Castleman, K. Second edn. Prentice-Hall, Englewood Clliffs, New Jersey (1996)
6. Pastor-Lopez, I., Santos, I., Santamaria-Ibirika, A., Salazar, M., de-la Pena-Sordo, J., Bringas, P.: Machine-learning-based surface defect detection and categorisation in high-precision foundry. In: Industrial Electronics and Applications (ICIEA), 2012 7th IEEE Conference on. (2012) 1359–1364
7. Heyer, L.J., Kruglyak, S., Yooseph, S.: Exploring expression data: identification and analysis of coexpressed genes. Genome research **9**(11) (1999) 1106–1115
8. Ugarte-Pedrero, X., Santos, I., Bringas, P., Gastesi, M., Esparza, J.: Semi-supervised learning for packed executable detection. In: In Proceedings of the 5th International Conference on Network and System Security (NSS). (2011) 342–346

9. Gonzalez, R., Woods, R.: Digital image processing. Reading, Mass.: Addison-Wesley **16**(716) (1992)
10. Viola, P., Jones, M.: Robust real-time face detection. International journal of computer vision **57**(2) (2004) 137–154
11. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press (1995)
12. Chawla, N., Bowyer, K., Hall, L., Kegelmeyer, W.: SMOTE: synthetic minority over-sampling technique. Journal of Artificial Intelligence Research **16**(3) (2002) 321–357
13. Cooper, G.F., Herskovits, E.: A bayesian method for constructing bayesian belief networks from databases. In: Proceedings of the 1991 conference on Uncertainty in artificial intelligence. (1991)
14. Geiger, D., Goldszmidt, M., Provan, G., Langley, P., Smyth, P.: Bayesian network classifiers. In: Machine Learning. (1997) 131–163
15. Amari, S., Wu, S.: Improving support vector machine classifiers by modifying kernel functions. Neural Networks **12**(6) (1999) 783–789
16. Maji, S., Berg, A., Malik, J.: Classification using intersection kernel support vector machines is efficient. In: Proc. CVPR. Volume 1. (2008) 4
17. Üstün, B., Melssen, W., Buydens, L.: Visualisation and interpretation of support vector regression models. Analytica chimica acta **595**(1-2) (2007) 299–309
18. Cho, B., Yu, H., Lee, J., Chee, Y., Kim, I., Kim, S.: Nonlinear support vector machine visualization for risk factor analysis using nomograms and localized radial basis function kernels. IEEE Transactions on Information Technology in Biomedicine **12**(2) (2008) 247
19. Garner, S.: Weka: The Waikato environment for knowledge analysis. In: Proceedings of the 1995 New Zealand Computer Science Research Students Conference. (1995) 57–64
20. Quinlan, J.: C4. 5 programs for machine learning. Morgan Kaufmann Publishers (1993)
21. Breiman, L.: Random forests. Machine learning **45**(1) (2001) 5–32
22. Singh, Y., Kaur, A., Malhotra, R.: Comparative analysis of regression and machine learning methods for predicting fault proneness models. International Journal of Computer Applications in Technology **35**(2) (2009) 183–193