An Intelligent Image Processing System for Real-Time Detection of Surface Flaws

Yu-Cheng Chou^{1,*}, Wei-Chieh Liao², Ming Chang², and Po Ting Lin²

¹*Institute of Undersea Technology, National Sun Yat-sen University, Taiwan*

²*Department of Mechanical Engineering, Chung Yuan Christian University, Taiwan*

(Received 20 August 2015; Accepted 4 January 2016; Published on line 1 March 2016)

*Corresponding author: ycchou@mail.nsysu.edu.tw

DOI: [10.5875/ausmt.v6i1.1006](https://doi.org/10.5875/ausmt.v6i1.1006)

Abstract: This paper presents an intelligent parallel image processing system, PVCS (Parallel Visual Computing System), to handle in-line surface defect detection for large objects. PVCS is a parallel expansion of a previously-reported non-destructive Compute Unified Device Architecture (CUDA)-enabled optical inspection system to accommodate large test objects. PVCS is heterogeneous both in terms of hardware and software. From the hardware perspective, PVCS consists of multiple CPUs and GPUs; from the software perspective, PVCS adopts the Message Passing Interface (MPI) and CUDA programming models. A parallel prototype system, consisting of three CPUs and two GPUs, is used to inspect a simulated object with an area eight times greater than that of the real object in our previous work. Given the same resolution requirements, the simulation results show that PVCS can obtain the correct number of defects within a large-size image at a satisfactory processing rate.

Keywords: intelligent image processing, surface flaw detection, parallel computing, Compute Unified Device Architecture (CUDA), Message Passing Interface (MPI)

Introduction

The manufacturing industry has adopted machine vision technology as a non-destructive test protocol for the reliable inspection and localization of surface defects. Machine vision technology relies on image analysis to provide useful and effective functions, such as automated optical inspection, remote navigation, and dynamic visual recognition. Over the past decade, numerous machine vision techniques have been proposed for surface flaw inspection purposes. Some recently reported methods and systems are summarized as follows. A machine vision system with a line scan camera was designed to detect differences in texture and brightness between well-scarfed and poorly-scarfed steel slab surfaces (Ryu et al. 2014). A line scan color vision system was designed to detect printing flaws on the color surfaces of drinking glasses decorated in a silk-screen process (Busin et al. 2013). An image recognition based vision system was developed to identify damage to engineering ceramic

grinding surfaces (Chen et al. 2013). A regularity measure was proposed as the only discrimination feature to detect ill-defined subtle defects on non-textured and homogeneously textured surfaces (Tsai et al. 2012). A local annular contrast based image processing algorithm was proposed to find defects on steel bar surfaces (Li et al. 2012). A dissimilarity measure based on the optical-flow technique was proposed to inspect defects on light-emitting diode (LED) wafer die surfaces (Tsai et al. 2012). A scanning system based on laser triangulation and machine vision technologies was established for surface defect inspection in continuous slab casting (Zhao et al. 2011). A multi-class support vector machine (SVM) based vision system was designed for defect inspection on strongly reflective metal surfaces (Zhang et al. 2011). An in-line inspection algorithm was proposed for textured plastic surfaces produced in a high-speed continuous process (Michaeli and Berdel 2011). A real-time defect detection system was designed for highly reflective curved surfaces of coated plastic components produced in the automotive industry (Rosati et al. 2009). Some machine

vision techniques were proposed to inspect micro-crack defects in the solar wafer manufacturing process (Chiou et al. 2011).

For screening products based on pre-defined criteria, the above techniques satisfactorily detect surface defects in different applications, but require a tradeoff between speed and resolution. Our previous work (Chang et al. 2014) presented a non-destructive optical inspection system that combines a Compute Unified Device Architecture (CUDA) (Lindholm et al. 2008)-enabled workstation to achieve satisfactory extraction and labeling of micro-sized surface defects. However, that study did not address surface defect detection for larger objects. The present paper presents an intelligent parallel image processing system to achieve in-line, high-speed and high-resolution surface defect detection.

The presented system is termed as PVCS (Parallel Visual Computing System). PVCS consists of independent machines that together form a master-slave computing model. Slave machines perform the same computations on different data in parallel; the master machine integrates all the results from the slave machines and, if desired, performs further analysis on the integrated result. An image to be processed by a slave machine during each time window typically contains hundreds of megapixels, and the time windows range from one to several seconds. PVCS represents the parallel expansion of our non-destructive CUDA-enabled optical inspection system, with the aim to handle in-line surface flaw detection for large objects.

Yu-Cheng Chou is an Assistant Professor in the Institute of Undersea Technology at National Sun Yat-sen University, Taiwan. Dr. Chou received his Ph.D. degree in Mechanical and Aeronautical Engineering from the University of California, Davis, USA, in 2009. His research interests include the development of intelligent control systems and design optimization schemes for underwater vehicles and instruments, multi-agent system design, distributed computing, and artificial intelligence optimization.

Wei-Chieh Liao received his master degree in Mechanical Engineering from Chung Yuan Christian University, Taiwan, in 2014.

Ming Chang is a Professor in the Department of Mechanical Engineering at Chung Yuan Christian University, Taiwan. Dr. Chang received his Ph.D. degree in Mechanical Engineering from National Taiwan University in 1986. He published more than one hundred SCIE and EI journal papers and three books. He is currently the director of Chinese Metrology Society and vice president of Taiwan Automatic Optical Inspection Equipment Association. He is conducting an extensive research program in nano-fabrication, especially in the construction of functional nano devices and equipment with assembling nano materials and components for applications in biomedical engineering.

Po Ting Lin is an Associate Professor in the Department of Mechanical Engineering at Chung Yuan Christian University, Taiwan. He is also with Research Center for Microsystem Engineering and Center for Robotics Research at Chung Yuan Christian University. Dr. Lin received his Ph.D. degree in Mechanical Engineering from Rutgers, The State University of New Jersey, USA, in 2010. His research interests include simulation and design of thermal systems, mechanical design optimization, reliability-based design optimization, multidisciplinary design optimization, robotics, and robot vision.

The rest of the paper is organized as follows. Section 2 describes the hardware and software architectures of PVCS. Section 3 illustrates the CUDA-based defect detection algorithms. Section 4 presents a defect detection test with a simulated object. Section 5 concludes the paper.

System Architecture

PVCS is heterogeneous both in terms of hardware and software.

Hardware Architecture

The hardware architecture in our previous work (Chang et al. 2014) has features a single CUDA workstation with a line scan camera. The PVCS hardware architecture of the PVCS is a parallel expansion of the previous one in our previous work. As shown in Fig. 1, the PVCS hardware architecture of the PVCS consists of independent machines that form a master-slave parallel computing model. The integration server is the master machine and each CUDA workstation is a slave machine. Each CUDA workstation performs the same computations and sends the result to the integration server, where for the integration and further analysis of the results will be conducted. All the machines are connected through a high-speed network. In addition, each CUDA platform has at least one CPU and one GPU. Two types of processing components, including CPU and GPU, are combined in the PVCS to achieve for parallel computation, thus the purposes. Thereby, the PVCS hardware is heterogeneous in hardware.

In the PVCS, depending on the test object's length and width, and the requirements on the resolution and speed requirements, PVCS configures different numbers of CUDA workstations can be configured to perform the computations in parallel. In Fig. 1, each color block represents the image data saved in a line scan camera's onboard memory in each time window. After a time window elapses, the image data is sent from a line scan camera's onboard memory to the main memory of a CUDA workstation's main memory. Thus, each color block represents the image data to be processed by a CUDA workstation in each time window before the new image data arrives. In Fig. 1, a 10-block image in one color stands for 1/5 of the entire object image under a specific resolution requirement. A single CUDA workstation ideally needs to spend requires 50 time windows to process the entire object image for an , if the object is carried by on a platform capable of moving in the X and Y directions. By using five properly configured CUDA workstations, the entire object image can be processed in 10 time windows.

Figure 1. PVCS hardware architecture.

Software Architecture

As shown in Fig. 2, the PVCS software architecture consists of MPI (Message Passing Interface) processes (Gropp et al. 1996; Gropp et al. 1999; Gropp et al. 1999) and CUDA threads (Lindholm et al. 2008; NVIDIA 2014). Because MPI processes run in CPUs and CUDA threads run in GPUs, PVCS software is heterogeneous.

MPI processes include master and slave processes. The MPI master process runs on the integration server, whereas the MPI slave processes run on the CUDA workstations. PVCS lets the MPI slave processes handle defect detection while the MPI master process deals with defect classification. Therefore, each MPI slave process launches a huge number of CUDA threads that carry out the defect detection procedures. As shown in Fig. 2, each MPI slave process will iteratively capture an image, launch CUDA threads to detect defects, store the processed image in a folder shared by all the MPI processes, and perform MPI process synchronization. Each CUDA thread, launched by a slave process, will binarize the image, label defects in the image, calculate the number of defects, and detect the defect edges. On the other hand, the MPI master process will iteratively perform MPI process synchronization, access the images stored in the shared folder, and classify the defects. The MPI process synchronization ensures that in each time window, the master process can use complete and correct images from the slave processes for defect classification.

In addition, the integration server can also be expanded as a cluster computer, which consists of multiple off-the-shelf computers, to increase computational efficiency for defect classification. Under such conditions, another group of MPI slave processes can be configured to perform CPU-based defect classification in parallel.

CUDA-Based Defect Detection Algorithms

Defect detection includes pixel-wise operations. Thus, in PVCS, each CUDA thread performs binarization, labeling, and edge detection operations on each pixel.

In a CUDA workstation, the MPI slave process needs to directly load the real-time image data into the memory accessible by the CPU, and then send the image data to the memory accessible by all the CUDA threads. In other words, for the CUDA threads to perform the pixel-wise operations, the image data needs to be transferred from the computer's main memory to the CUDA GPU's global memory. After the CUDA threads complete their operations, the processed image data must be transferred from the CUDA GPU's global memory back to the computer's main memory. Thus, these two data transfers between the CPU and GPU are mandatory.

The binarization, labeling, and edge detection algorithms, which are carried out by the CUDA threads, are presented in the following sub-sections.

Figure 2. PVCS software architecture.

Binarization

The first step in defect detection is to binarize an image. As shown in Fig. 3, the binarization is straightforward and based on a predefined threshold. If a pixel value is larger than the threshold, the pixel value is set to 255. Otherwise, the pixel value is set to zero.

```
BEGIN
1 IF pixel value > threshold THEN
2   pixel value ← 255
  ELSE
3   pixel value ← 0
END
```

Figure 3. Binarization.

Labeling

Determining the number of defects requires labeling all the defects in an image. Labeling involves finding the starting and terminal points of each defect, and then eliminating the redundant starting and terminal points to derive the total number of defects.

The defect starting point determination is illustrated in Fig. 4. For a pixel value equal to zero, the pixel value is not changed. For a pixel value larger than zero, if its left, upper-left, upper, and upper-right pixel values are all zeroes, the pixel value is set to 1; otherwise, the pixel value is set to 9. Pixels with a value of 1 are the defect starting points.

The defect terminal point determination is illustrated in Fig. 5. For a pixel value equal to zero or 1, the pixel value is not changed. For a pixel value larger than zero and not equal to 1, if its right, bottom-right, bottom, and bottom-left pixel values are all zeroes, the pixel value is set to 2; otherwise, the pixel value is set to 9. Pixels with a value of 2 are the defect terminal points.

The redundant defect starting point elimination is illustrated in Fig. 6. This operation targets pixels with a value of 1. For such a target pixel, if any pixel with a value of 1 exists to its left in the same row, the target pixel value is set to 9.

The redundant defect terminal point elimination is illustrated in Fig. 7. This operation targets pixels with a value of 2. For such a target pixel, if any pixel with a value of 2 exists to its right in the same row, the target pixel value is set to 9.

Edge Detection

Edge detection obtains the defect contours, i.e., the defect's left, upper, right, and bottom edges. For edge detection, any nonzero pixel value due to the labeling operation will be changed back to 255, which is the value immediately following the binarization operation. Moreover, an array the same size as the test image is

allocated and initialized with zeros in the GPU's global memory for edge detection.

```
BEGIN
1 IF pixel value > 0 THEN
2   IF left pixel value = 0 THEN
3     IF upper left pixel value = 0 THEN
4       IF upper pixel value = 0 THEN
5         IF upper right pixel value = 0 THEN
6           pixel value ← 1
        ELSE
7           pixel value ← 9
        ELSE
8           pixel value ← 9
        ELSE
9           pixel value ← 9
        ELSE
10          pixel value ← 9
END
```

Figure 4. Defect starting point determination.

```
BEGIN
1 IF pixel value > 0 AND pixel value ≠ 1 THEN
2   IF right pixel value = 0 THEN
3     IF bottom right pixel value = 0 THEN
4       IF bottom pixel value = 0 THEN
5         IF bottom left pixel value = 0 THEN
6           pixel value ← 2
        ELSE
7           pixel value ← 9
        ELSE
8           pixel value ← 9
        ELSE
9           pixel value ← 9
        ELSE
10          pixel value ← 9
END
```

Figure 5. Defect terminal point determination.

```
BEGIN
1 IF pixel value = 1 THEN
2   IF column index of left pixel ≥ boundary limit THEN
3     IF pixel value = left pixel value THEN
4       pixel value ← 9
    ELSE
5     DECREMENT column index of left pixel
    GOTO 2
END
```

Figure 6. Redundant defect starting point elimination.

```
BEGIN
1 IF pixel value = 2 THEN
2   IF column index of right pixel ≤ boundary limit THEN
3     IF pixel value = right pixel value THEN
4       pixel value ← 9
    ELSE
5     INCREMENT column index of right pixel
    GOTO 2
END
```

Figure 7. Redundant defect terminal point elimination.

The defect left edge detection is shown in Fig. 8. This operation targets pixels with a value of 255. For such a target pixel, if its left pixel value is zero and right pixel

value is 255, its corresponding pixel value in the new array is set to 255; otherwise, its corresponding pixel value in the new array is not changed.

The defect upper edge detection is shown in Fig. 9. This operation targets pixels with a value of 255. For such a target pixel, if its upper pixel value is zero, its corresponding pixel value in the new array is zero, and its bottom pixel value is 255, its corresponding pixel value in the new array is set to 255; otherwise, its corresponding pixel value in the new array is not changed.

The defect right edge detection is shown in Fig. 10. This operation targets pixels with a value of 255. For such a target pixel, if its right pixel value is zero and left pixel value is 255, its corresponding pixel value in the new array is set to 255; otherwise, its corresponding pixel value in the new array is not changed.

The defect bottom edge detection is shown in Fig. 11. This operation targets pixels with a value of 255. For such a target pixel, if its bottom pixel value is zero, its corresponding pixel value in the new array is zero, and its upper pixel value is 255, its corresponding pixel value in the new array is set to 255; otherwise, its corresponding pixel value in the new array is not changed.

```
BEGIN
1 IF pixel value > 0 THEN
2   IF left pixel value > 0 THEN
3     IF right pixel value = 255 THEN
4       pixel value in the new array ← 255
5     END
6   END
7 END
```

Figure 8. Defect left edge detection.

```
BEGIN
1 IF pixel value > 0 THEN
2   IF upper pixel value = 0 THEN
3     IF pixel value in the new array = 0 THEN
4       IF bottom pixel value = 255 THEN
5         pixel value in the new array ← 255
6       END
7     END
8   END
9 END
```

Figure 9. Defect upper edge detection.

```
BEGIN
1 IF pixel value > 0 THEN
2   IF right pixel value = 0 THEN
3     IF left pixel value = 255 THEN
4       pixel value in the new array ← 255
5     END
6   END
7 END
```

Figure 10. Defect right edge detection.

```
BEGIN
1 IF pixel value > 0 THEN
2   IF bottom pixel value = 0 THEN
3     IF pixel value in the new array = 0 THEN
4       IF upper pixel value = 255 THEN
5         pixel value in the new array ← 255
6       END
7     END
8   END
9 END
```

Figure 11. Defect bottom edge detection.

Simulated Test Results

This section presents a proof-of-concept defect detection simulated test using a parallel prototype of PVCS.

Hardware and Software Configurations

As shown in Fig. 12, one integration server and two CUDA workstations are used in this experiment. In addition, an MPI master process (Process 0) runs on the integration server, and each of the two MPI slave processes (Process 1 and Process 2) runs on a CUDA workstation. In this experiment, the MPI master process does not perform any calculations and simply receives the number of defects from each MPI slave process. Each MPI slave process performs the binarization, labeling and edge detection through CUDA threads.

In our previous work, the resolution requirement is that each pixel represents an area of $3.5 \times 3.5 \mu\text{m}^2$. The camera is a 12288-pixel line scan camera with a 12 kHz acquisition rate and a 512 MB onboard memory. The largest mirror object, for which the whose image size is affordable can be accommodated by the camera's onboard memory, is 70 mm in length and 43 mm in width. Based on the camera's hardware specifications, the speed requirement is specified such that the surface inspection of the mirror object needs to must be completed within 2 seconds, equivalent to a processing rate of 122.88 mega pixels per second.

Under the same resolution requirement as that in our previous work, this defect detection test simulates a situation where in which an object, of measuring 280 mm in length and 86 mm in width, is examined using two CUDA workstations equipped with line scan cameras. Therefore, the object's entire image of the object has measures 80,000 pixels in length and 24,576 pixels in width. Under the same speed requirement (i.e. 122.88 mega pixels per second), the entire image has to be processed within 8 seconds using this hardware setup.

Due to the maximum global memory on the GPU, the entire image has to be divided into 32 test images. Each test image has 5,000 pixels in length and 12,288 pixels in width, and it represents an area of $1.75 \times 4.3 \text{ cm}^2$, as shown in Fig. 12. A total of 16 test images are successively read and processed by each CUDA workstation.

Detection Results

Table 1 shows the detection results from the first CUDA workstation. For each test image, the number of defects obtained (44) is correct and the total processing time is 2,726.88 milliseconds, which is 34.09% of the speed requirement (8,000 milliseconds), equivalent to a

Figure 12. Configurations of the defect detection experiment.

processing rate of 360.50 mega pixels per second.

Table 2 shows the detection results from the second CUDA workstation. For each test image, the number of defects obtained (44) is correct and the total processing time is 3,084.64 milliseconds, which is 38.56% of the speed requirement, equivalent to a processing rate of 318.69 mega pixels per second.

Results Image	Time (milliseconds)	Number of defects
1	171.06	3
2	170.64	2
3	172.01	3
4	171.33	3
5	169.74	3
6	171.55	2
7	172.74	4
8	170.33	2
9	168.56	1
10	159.74	2
11	172.31	3
12	168.67	3
13	169.71	3
14	172.05	4
15	174.23	3
16	172.21	3
Total	2,726.88	44

Table 1. Results from the 1st CUDA workstation (MPI Process 1).

From Tables 1 and 2 we know that, under the same resolution and speed requirements as those in our previous work, for a simulated object with an image size

eight times the capacity of the line scan camera's onboard memory, PVCS can obtain the correct number of defects on the object. The processing times used by the two CUDA workstations are 34.09% and 38.56% of the speed requirement, corresponding to processing rates of 360.50 and 318.69 mega pixels per second.

Results Image	Time (milliseconds)	Number of defects
1	217.95	2
2	204.60	3
3	190.14	3
4	191.24	3
5	188.65	2
6	192.82	3
7	189.92	3
8	191.29	4
9	194.93	2
10	206.33	3
11	190.54	2
12	183.79	3
13	186.03	3
14	187.36	3
15	187.87	4
16	181.18	1
Total	3,084.64	44

Table 2. Results from the 2nd CUDA workstation (MPI Process 2).

Conclusion

This paper presents an intelligent parallel image processing system, Parallel Visual Computing System

(PVCS), a parallel expansion of our previously described non-destructive CUDA enabled optical inspection system, to handle in-line surface defect detection for large objects. PVCS has an expandable heterogeneous hardware architecture consisting of multiple CPUs and GPUs. In addition, PVCS has a heterogeneous software architecture incorporating MPI and CUDA programming models. PVCS is validated through a defect detection simulated test using a PVCS prototype consisting of two CUDA workstations equipped with line scan cameras. The simulated test results show that, under the same resolution and speed requirements as those in our previous work, for a simulated object with an image size eight times the capacity of the line scan camera's onboard memory, PVCS can obtain the correct number of defects on the object. The processing times used by the two CUDA workstations are 34.09% and 38.56% of the speed requirement, equivalent to processing rates of 360.50 and 318.69 mega pixels per second, respectively. Therefore, PVCS shows good accuracy and computational speed for in-line surface defect detection of large objects.

Acknowledgement

This research was supported by the Ministry of Science and Technology, Taiwan, under grants MOST104-2221-E-110-062, MOST103-2221-E-110-087, and NSC102-2218-E-033-002-MY2.

References

- [1] L. Busin, N. Vandenbroucke and L. Macaire, "Contribution of a color space selection to a flaw detection vision system", *Journal of Electronic Imaging*, vol. 22, no. 3, p. 033016, 2013.
doi: [10.1117/1.jei.22.3.033016](https://doi.org/10.1117/1.jei.22.3.033016)
- [2] M. Chang, Y.-C. Chou, P.-T. Lin, and J. L. Gabayno, "Fast and high-resolution optical inspection system for in-line detection and labeling of surface defects," *CMC-Computers Materials & Continua*, vol. 42, pp. 125–140, 2014.
- [3] S. Chen, B. Lin, X. Han, and X. Liang, "Automated inspection of engineering ceramic grinding surface damage based on image recognition," *The International Journal of Advanced Manufacturing Technology*, vol. 66, no. 1–4, pp. 431–443, 2013.
doi: [10.1007/s00170-012-4338-2](https://doi.org/10.1007/s00170-012-4338-2)
- [4] Y.-C. Chiou, J.-Z. Liu, and Y.-T. Liang, "Micro crack detection of multi-crystalline silicon solar wafer using machine vision techniques," *Sensor Review*, vol. 31, no. 2, pp. 154–165, 2011.
doi: [10.1108/02602281111110013](https://doi.org/10.1108/02602281111110013)
- [5] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A high-performance, portable implementation of the MPI message passing interface standard," *Parallel Computing*, vol. 22, no. 6, pp. 789–828, 1996.
- [6] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: Portable Parallel Programming with the Message-passing Interface*. MIT Press, 1999.
- [7] W. Gropp, E. Lusk, and R. Thakur, *Using MPI-2: Advanced Features of the Message-passing Interface*. MIT Press, 1999.
- [8] W.-B. Li, C.-H. Lu, and J.-C. Zhang, "A local annular contrast based real-time inspection algorithm for steel bar surface defects," *Applied Surface Science*, vol. 258, no. 16, pp. 6080–6086, 2012.
doi: [10.1016/j.apsusc.2012.03.007](https://doi.org/10.1016/j.apsusc.2012.03.007)
- [9] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym, "NVIDIA Tesla: A Unified Graphics and Computing Architecture," *IEEE Micro*, vol. 28, no. 2, pp. 39–55, 2008.
- [10] W. Michaeli and K. Berdel, "Inline inspection of textured plastics surfaces," *Optical Engineering*, vol. 50, no. 2, p. 027205, 2011.
doi: [10.1117/1.3544588](https://doi.org/10.1117/1.3544588)
- [11] *CUDA C Programming Guide*, NVIDIA, Santa Clara, CA, 2015.
<http://docs.nvidia.com/cuda/cuda-c-programming-guide/>
- [12] G. Rosati, G. Boschetti, A. Biondi, and A. Rossi, "Real-time defect detection on highly reflective curved surfaces," *Optics and Lasers in Engineering*, vol. 47, no. 3, pp. 379–384, 2009.
doi: [10.1016/j.optlaseng.2008.03.010](https://doi.org/10.1016/j.optlaseng.2008.03.010)
- [13] S.-G. Ryu, D.-C. Choi, Y.-J. Jeon, S.-J. Lee, J.-P. Yun, and S.-W. Kim, "Detection of Scarfing Faults on the Edges of Slabs," *ISIJ International*, vol. 54, no. 1, pp. 112–118, 2014.
doi: [10.2355/isijinternational.54.112](https://doi.org/10.2355/isijinternational.54.112)
- [14] D.-M. Tsai, M.-C. Chen, W.-C. Li, and W.-Y. Chiu, "A fast regularity measure for surface defect detection," *Machine Vision and Applications*, vol. 23, no. 5, pp. 869–886, 2012.
doi: [10.1007/s00138-011-0403-3](https://doi.org/10.1007/s00138-011-0403-3)
- [15] D.-M. Tsai, I.-Y. Chiang, and Y.-H. Tsai, "A Shift-Tolerant Dissimilarity Measure for Surface Defect Detection," *Transactions on Industrial Informatics*, vol. 8, no. 1, pp. 128–137, 2012.
doi: [10.1109/tii.2011.2166797](https://doi.org/10.1109/tii.2011.2166797)
- [16] X.-W. Zhang, Y.-Q. Ding, Y.-Y. Lv, A.-Y. Shi, and R.-Y. Liang, "A vision inspection system for the surface defects of strongly reflected metal based on multi-class SVM," *Expert Systems with Applications*, vol. 38, no. 5, pp. 5930–5939, 2011.
doi: [10.1016/j.eswa.2010.11.030](https://doi.org/10.1016/j.eswa.2010.11.030)
- [17] L.-M. Zhao, Q.-Ouyang, D.-F. Chen, and L.-Y. Wen,

“Surface defects inspection method in hot slab continuous casting process,” *Ironmaking &*

Steelmaking, vol. 38, no. 6, pp. 464–470, 2011.
doi: [10.1179/1743281211y.0000000025](https://doi.org/10.1179/1743281211y.0000000025)