In-Line Detection of Surface Profile and Defects Using Full-Field and High-Precision Automated Optical Inspection with Cloud-Based Analysis System

Ming Chang, Yu-Cheng Chou, Po Ting Lin

Department of Mechanical Engineering
Chung Yuan Christian University
Chungli, Taoyuan, Taiwan
Ming Chang, Yu-Cheng Chou, Po Ting Lin

Department of Mechanical Engineering
Chung Yuan Christian University
Chungli, Taoyuan, Taiwan

Optical Measurement
Optomechatronics
Automated Optical Inspection

Parallel Computing
Cloud Computing
Software Design
Intelligent Embedded System

Design Under Uncertainty
Multidisciplinary Design Optimization
Multiobjective Optimization
Inspection of Handheld Devices

All images are from google.com.
Full-field In-Line Inspection System

- Line CCD module
- Linear light source
- Automated inspection line
- GPU module
- Control Center
- GPU#1
- GPU#2
- Parallel defect inspection
- GPU#N
- Flawless products
- Converyer
- Defect
- Post-processing

Parallel image processing

- GPU module
- Control Center
2D Inspection of Defect in Back-coated Glass
Motivation and Objectives

- Numerous methods have been proposed for different applications of surface defect detection, and those methods can generate satisfactory performance.
- However, those methods have not addressed a situation where both the speed and precision requirements need to be satisfied simultaneously.
- In other words, an image to be processed by a machine during each time window has hundreds of mega pixels, whereas the time window is within one second.
- This presentation shows a highly expandable distributed image sensor computing system, DISCS, to achieve in-line surface defect detection with high performance on both the speed and precision.
(a) Hardware components of the optical inspection platform.
(b) Actual photo of one opto-mechanical module.
(c) Construction of illumination device with two 24-LED arrays
Inspection Principle

Glass substrate
Scattering effect at reflective coating induced by defect
Line Scanned Camera
High reflective coating
Defect
Slanted parallel light source
Moving direction
Slanted angle

Fan-shape lighting of LED source:
Form the side-lighting effect consequently
Light slit
Position of image acquisition
High reflective coating

Position of image acquisition
Moving direction
Distributed Image Sensor Computing System (DISCS)

- **Characteristics**
  - Heterogeneous parallel computing system
  - Consists of multiple CPUs and GPUs
  - Adopts Message Passing Interface (MPI) and Compute Unified Device Architecture (CUDA) programming models
  - High speed and high precision in-line detection of surface defects

- **Hardware Development**
  - Consists of independent machines that form a master-slave parallel computing model.
  - Each CUDA workstation is a slave machine, which performs the same computations and sends the result to the master machine and has at least one CPU and one GPU.
  - All the machines are connected through a high-speed network.
Hardware architecture of the DISCS
System Architecture - Software

- Consists of MPI processes and CUDA threads.
- MPI processes run in CPUs and CUDA threads run in GPUs.
- The MPI master process runs on the integration server, whereas the MPI slave processes run on the CUDA workstations.
- The idea of the DISCS is to let the MPI slave processes handle the defect detection, and let the MPI master process deal with the defect classification.

Software architecture of the DISCS.
CUDA-Based Defect Detection Algorithms - Binarization

- 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.

The binarization algorithm.
CUDA-Based Defect Detection Algorithms - Labeling

**Situation 1**
- Target pixel's value > 0
- Target pixel's left, upper-left, upper, and upper-right pixel value = 0

**Situation 2**
- Target pixel's value > 0
- Target pixel's left, upper-left, and upper-right pixel value = 0, but upper pixel value ≠ 0

**Determination of Starting Point**
CUDA-Based Defect Detection Algorithms - Labeling

**Situation 1**
- Target pixel value = 9
- Target pixel's right, bottom-right, bottom, and bottom-left pixel value = 0

**Situation 2**
- Target pixel value = 9
- Target pixel's right and bottom-left pixel value = 0, but bottom-right and bottom pixel value ≠ 0

**Determination of End Point**
The target pixel value = 1

If there is any pixel, which is on the target left side in the same row with 10 pixel width and has a value of 1

The target pixel’s value is changed from 1 to 9, otherwise do nothing

Elimination of Redundant Starting Point
CUDA-Based Defect Detection Algorithms - Labeling

The target pixel value = 2

if there is any pixel, which is on the target right side in the same row with 10 pixel width and has a value of 2

The target pixel’s value is changed from 1 to 9, otherwise do nothing

Elimination of Redundant End Point
### CUDA-Based Defect Detection Algorithms – Edge Detection

#### Detection of Upper-Left Edge

<table>
<thead>
<tr>
<th>Array</th>
<th>Target pixel’s left-upper pixel value = 0 &amp; Target pixel’s right-bottom pixel value = 255</th>
<th>Array A</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Array" /></td>
<td><img src="image2" alt="Target pixel’s left-upper pixel value = 0 &amp; Target pixel’s right-bottom pixel value = 255" /></td>
<td><img src="image3" alt="Array A" /></td>
</tr>
</tbody>
</table>

- **No action**

**Detection of Upper-Left Edge**
CUDA-Based Defect Detection Algorithms – Edge Detection

**Target pixel’s upper pixel value = 0**

& **Target pixel’s bottom pixel value = 255**

& **Array A’s target pixel value ≠ 255**

<table>
<thead>
<tr>
<th>Array</th>
<th>Array A</th>
<th>No action</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Array Image" /></td>
<td><img src="image2.png" alt="Array A Image" /></td>
<td><img src="image3.png" alt="No action Image" /></td>
</tr>
</tbody>
</table>
CUDA-Based Defect Detection Algorithms – Edge Detection

Target pixel's right-upper pixel value = 0
& Target pixel's left-bottom pixel value = 255
& Array A's target pixel value ≠ 255

Detection of Upper-Right Edge
CUDA-Based Defect Detection Algorithms – Edge Detection

Target pixel's right pixel value = 0
& Target pixel's left pixel value = 255
& Array A's target pixel value ≠ 255

No action

Detection of Right Edge
CUDA-Based Defect Detection Algorithms – Edge Detection

<table>
<thead>
<tr>
<th>Array</th>
<th>Target pixel's right-bottom pixel value = 0 &amp; Target pixel's left-upper pixel value = 255 &amp; Array A’s target pixel value ≠ 255</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0</td>
<td>0 0 0 255 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 255 255 255 0 0</td>
<td>0 0 0 255 255 255 0 0</td>
</tr>
<tr>
<td>0 0 255 255 255 255 0 0</td>
<td>0 255 255 255 255 0 0</td>
</tr>
<tr>
<td>0 0 255 255 255 255 0 0</td>
<td>0 255 255 255 0 0</td>
</tr>
<tr>
<td>0 0 255 255 255 0 0 0</td>
<td>0 255 255 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Array A</th>
<th>No action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0</td>
<td>0 0 0 255 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 255 255 255 0 0</td>
<td>0 0 0 255 255 255 0 0</td>
</tr>
<tr>
<td>0 0 255 255 255 255 0 0</td>
<td>0 255 255 255 255 0 0</td>
</tr>
<tr>
<td>0 0 255 255 255 255 0 0</td>
<td>0 255 255 255 0 0</td>
</tr>
<tr>
<td>0 0 255 255 255 0 0 0</td>
<td>0 255 255 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

Target pixel's right-bottom pixel value = 0 & Target pixel's left-upper pixel value = 255 & Array A’s target pixel value ≠ 255

Detection of Lower-Right Edge
CUDA-Based Defect Detection Algorithms – Edge Detection

Target pixel's bottom pixel value = 0
& Target pixel's upper pixel value = 255
& Array A's target pixel value ≠ 255

No action

Detection of Lower Edge
CUDA-Based Defect Detection Algorithms – Edge Detection

Array

Target pixel’s left-bottom pixel value = 0
& Target pixel’s right-upper pixel value = 255
& Array A’s target pixel value ≠ 255

Array A

Detection of Lower-Left Edge
CUDA-Based Defect Detection Algorithms – Edge Detection

Target pixel's left pixel value = 0 & Target pixel's right pixel value = 255 & Array A's target pixel value ≠ 255

Detection of Left Edge
CUDA-Based Defect Detection Algorithms - Redundancy Detection

Test object's left part

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>255</th>
<th>255</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Test object's right part

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>255</th>
<th>255</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Array C

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>255</th>
<th>255</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Array B

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>255</td>
</tr>
<tr>
<td>255</td>
<td>255</td>
</tr>
</tbody>
</table>
CUDA-Based Defect Detection Algorithms - Redundancy Detection

Array B

Array C

Array B

Array B

Array B

Array B

Array B

Array B

Array B
Complete Process Flow of 2D Defect Inspection

Start → parameters setting → System Initialize → Acquire Image → Image Pre-processing → Defect Detection → end

Host code (CPU):
- GPU Memory Allocation:
  - raw → dst_1, dst_2, dst_3, dst_4, dst_5, dst_out
- Copy Memory:
  - Host to Device:
    - raw = Image
- Design Block and Thread for:
  - Histogram Equalization:
    - Start kernel
  - Gaussian Pyramids:
    - Start kernel
- Design Block and Thread for:
  - Binarization:
    - Start kernel
- Design Block and Thread for:
  - Label:
    - Start kernel
- Design Block and Thread for:
  - Size Detection:
    - Start kernel
- Design Block and Thread for:
  - Edge Detection:
    - Start kernel
- Copy Memory:
  - Device to Host:
    - Image = dst_out

Device code (GPU):
- raw →
  - Histogram Equalization:
    - dst_1
  - Gaussian Pyramids:
    - dst_2
  - Binarization:
    - dst_3
  - Label:
    - dst_4
  - Size Detection:
    - dst_5
  - Edge Detection:
    - dst_out
Experimental Results of DISCS

- The hardware configuration of the experiment includes a test object and two CUDA workstations, which controls a line scan camera.
- In the experiment, CUDA C program and MPI functions were used on the CUDA workstation.
- The test object’s width and length are 8.6 cm and 28 cm, respectively.
- The precision requirement specifies that each image pixel represents a 3.5 μm x 3.5 μm area.
- Each time window needs to be calculated within 250 ms.
- In each time window, the CUDA workstation needs to finish the defect detection in an image of 12288 x 5000 pixels.
- Totally 16 image strips are to be inspected within 4 s (70 mm/s).
Experimental Results of DISCS

- Dark-field image of a part of one back-coated mirror piece

<table>
<thead>
<tr>
<th>Features</th>
<th>Average size</th>
<th>Detection Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bubbles</td>
<td>Point-like or circular shape</td>
<td>50 to 160 μm</td>
</tr>
<tr>
<td>Cracks</td>
<td>Thin, elongated, located far from the perimeter of the test object</td>
<td>Width: 10 to 20 μm</td>
</tr>
<tr>
<td>Edge defects</td>
<td>Jagged lines, voids</td>
<td>Length: 1 to 10 mm</td>
</tr>
</tbody>
</table>
## Experimental Results of Simulated Defect Patterns (1)

<table>
<thead>
<tr>
<th>Time</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>12288 × 5000 pixels</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU to GPU (ms)</td>
<td>59.296</td>
</tr>
<tr>
<td>CUDA kernel function (ms)</td>
<td>47.3</td>
</tr>
<tr>
<td>GPU to CPU (ms)</td>
<td>57.39</td>
</tr>
<tr>
<td>Total time (ms)</td>
<td>163.986</td>
</tr>
<tr>
<td>Defect number</td>
<td>4</td>
</tr>
</tbody>
</table>
Experimental Results of Simulated Defect Patterns (2)

<table>
<thead>
<tr>
<th>Time</th>
<th>Size</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>12288 × 5000 pixels (left)</td>
<td>12288 × 5000 pixels (right)</td>
<td></td>
</tr>
<tr>
<td>CPU to GPU (ms)</td>
<td>61.0682</td>
<td>62.6171</td>
<td></td>
</tr>
<tr>
<td>CUDA kernel function (ms)</td>
<td>51.1604</td>
<td>51.2616</td>
<td></td>
</tr>
<tr>
<td>GPU to CPU (ms)</td>
<td>59.8082</td>
<td>59.6373</td>
<td></td>
</tr>
<tr>
<td>Redundant defect in CPU (ms)</td>
<td>0.1662</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Total time (ms)</td>
<td>172.203</td>
<td>173.516</td>
<td></td>
</tr>
<tr>
<td>Defect number</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>New defect number</td>
<td></td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
Experimental Results of Simulated Defect Patterns (3)

(a)  (b)  (c)  (d)
## Experimental Results of Simulated Defect Patterns (3)

<table>
<thead>
<tr>
<th>Time</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU to GPU (ms)</td>
<td>60.9912</td>
<td>58.3117</td>
<td>62.0705</td>
<td>61.5419</td>
</tr>
<tr>
<td>CUDA kernel function (ms)</td>
<td>51.4951</td>
<td>51.9195</td>
<td>52.8371</td>
<td>51.2708</td>
</tr>
<tr>
<td>GPU to CPU (ms)</td>
<td>60.5924</td>
<td>59.701</td>
<td>59.1734</td>
<td>60.3451</td>
</tr>
<tr>
<td>Redundant defect in CPU (ms)</td>
<td>0.1703</td>
<td>0.1847</td>
<td>0.1791</td>
<td>0</td>
</tr>
<tr>
<td>Total time (ms)</td>
<td>173.249</td>
<td>170.117</td>
<td>174.26</td>
<td>173.158</td>
</tr>
<tr>
<td>Defect number</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>New defect number</td>
<td></td>
<td></td>
<td></td>
<td>9</td>
</tr>
</tbody>
</table>
Experimental Results of Simulated Defect Patterns (4)

(a) (b) (c) (d)
## Experimental Results of Simulated Defect Patterns (4)

<table>
<thead>
<tr>
<th>Time</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU to GPU (ms)</td>
<td>61.0697</td>
<td>61.4254</td>
<td>61.9181</td>
<td>60.9409</td>
</tr>
<tr>
<td>CUDA kernel function (ms)</td>
<td>51.8225</td>
<td>50.9516</td>
<td>50.1037</td>
<td>52.3008</td>
</tr>
<tr>
<td>GPU to CPU (ms)</td>
<td>59.6589</td>
<td>58.9214</td>
<td>59.3474</td>
<td>59.5655</td>
</tr>
<tr>
<td>Redundant defect in CPU (ms)</td>
<td>0.03382</td>
<td>0.3402</td>
<td>0.1719</td>
<td>0</td>
</tr>
<tr>
<td>Total time (ms)</td>
<td>172.5849</td>
<td>171.6186</td>
<td>171.5411</td>
<td>172.8072</td>
</tr>
<tr>
<td>Defect number</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>New defect number</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Experimental Results of Simulated Defect Patterns (5)
Experimental Results of Simulated Defect Patterns (5)

<table>
<thead>
<tr>
<th>Time</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU to GPU (ms)</td>
<td>57.8416</td>
<td>61.0743</td>
<td>61.8852</td>
<td>61.4629</td>
</tr>
<tr>
<td>CUDA kernel function (ms)</td>
<td>52.1556</td>
<td>55.1779</td>
<td>50.9634</td>
<td>49.6541</td>
</tr>
<tr>
<td>GPU to CPU (ms)</td>
<td>56.3132</td>
<td>59.8729</td>
<td>60.3045</td>
<td>61.6009</td>
</tr>
<tr>
<td>Redundant defect in CPU (ms)</td>
<td>0.178</td>
<td>0.1914</td>
<td>0.1601</td>
<td>0</td>
</tr>
<tr>
<td>Total time (ms)</td>
<td>166.4884</td>
<td>176.3165</td>
<td>173.3132</td>
<td>172.7179</td>
</tr>
<tr>
<td>Defect number</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>New defect number</td>
<td></td>
<td></td>
<td></td>
<td>11</td>
</tr>
</tbody>
</table>
Experimental Results of Simulated Defect Patterns (6)
## Experimental Results of Simulated Defect Patterns (6)

<table>
<thead>
<tr>
<th>Time</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU to GPU (ms)</td>
<td>62.457</td>
<td>62.1429</td>
<td>62.2435</td>
<td>62.3949</td>
</tr>
<tr>
<td>CUDA kernel function (ms)</td>
<td>49.9108</td>
<td>48.172</td>
<td>51.9457</td>
<td>50.0745</td>
</tr>
<tr>
<td>GPU to CPU (ms)</td>
<td>97.956</td>
<td>95.8898</td>
<td>85.225</td>
<td>79.3337</td>
</tr>
<tr>
<td>Redundant defect in CPU (ms)</td>
<td>0.1765</td>
<td>0.1652</td>
<td>0.1693</td>
<td>0</td>
</tr>
<tr>
<td>Total time (ms)</td>
<td>210.5</td>
<td>206.3699</td>
<td>199.5835</td>
<td>191.8031</td>
</tr>
<tr>
<td>Defect number</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>New defect number</td>
<td></td>
<td></td>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>
Experimental Results of Simulated Defect Patterns (7)

- Total amount of time is calculated as 239.3 milliseconds and 209.91 milliseconds, which is within the time window, 250 milliseconds.
3D Inspection of Defect in Transparent Surface
Determination of 3D Profiles

- Projection fringe technique applies a straight-line grating onto an object to study the surface topography by recording the grating deformation due to topography variation.

- Shadow moiré topography positions the grating close to the object and the contour lines of the shadows at the object surface under the grating are observed.
Scanning Moiré Topography

- Scanning moiré technique, which is adapted from the traditional projection fringe technique, records contour images at the object surface using a linear CCD camera and a motorized transition stage.
- Similar to the conventional shadow moiré, the surface height distribution of the object is mathematically described as follows:

\[ h(x, y) = \frac{(x, y)}{2} P_0 \cot \alpha = N(x, y) P_n \]

- \( h(x, y) \): The object height at an arbitrary point \((x, y)\) relative to the virtual reference plane
- \( \phi(x, y) \): The phase difference distribution between object surface and reference plane for each point \((x, y)\)
- \( P_0 \) & \( P_n \): Respectively the grating pitch in the direction parallel and perpendicular to the reference plane
- \( \alpha \): The grating projection angle inclined to the optical axis of the CCD
- \( N(x, y) \): The fringe order of the surface contour at each point \((x, y)\)
Scanning Moiré Topography

- Scanning moiré technique, which is adapted from the traditional projection fringe technique, records contour images at the object surface using a linear CCD camera and a motorized transition stage.
- Similar to the conventional shadow moiré, the surface height distribution of the object is mathematically described as follows:

\[ h(x, y) = \frac{(x, y)}{2} P_0 \cot \alpha = N(x, y) P_n \]

- The surface contour is directly related:
  1. The phase distribution of the interferogram
  2. The measurement sensitivity of the object height is dependent on the projected grating pitch
  3. Grating incidence angle

The fringe order at any point is determined by the phase at that point so that the surface topography of the specimen can be extracted from its phase distribution. With an appropriate phase measuring technique, contour maps can be used to generate surface topography quantitatively.
Phase Measuring Technique

- Since the sets of contour maps will be obtained separately from the RGB channels, the inherent phase shift between each two of the three interferograms provides sufficient information for Phase Shifting Interferometry (PSI) on the fringe pattern.

\[ P = \frac{3S}{1 + 3n}, \quad n = 0, 1, 2... \]

\( P \): The grating image on the CCD \( S \): The pitches of the three RGB lines

- Three frames of intensity data were simultaneously recorded with a 120° phase change between any two adjacent readouts and are presented by

\[ I_1(x, y) = I_A + I_B \cos[\phi(x, y)] \]
\[ I_2(x, y) = I_A + I_B \cos[\phi(x, y) + 2n\pi + 120^0] \]
\[ I_3(x, y) = I_A + I_B \cos[\phi(x, y) + 4n\pi + 240^0] \]

- The phase distribution of the contour map is obtained:

\[ (x, y) = \tan^{-1} \frac{\sqrt{3(I_1 - I_3)}}{2I_2(I_1 + I_3)} \]

A continuous phase shift equivalent to 120° exists for the three sets of interferogram. For the red, green, and blue contour fringes, these correspond to 0°, +120°, and +240°, respectively.
Scanning Projection Fringe System

**System Specifications**

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution</td>
<td>4096 pixels</td>
</tr>
<tr>
<td>CCD Pixel Size</td>
<td>10 µm x 10 µm</td>
</tr>
<tr>
<td>The current magnitude</td>
<td>0.5x</td>
</tr>
</tbody>
</table>

The projected pitch of the grating on the CCD: 45 µm

120° phase shift between each color channel, the projected pitch of the grating on the CCD should be 22.5 µm for n = 1

\[ P = \frac{3S}{1+3n} \]
RGB Calibration

- Each channel has different photosensitivity.
- The RGB inputs to the CCD were calibrated by adjusting the output RGB lines of the DLP.

The projected grating image on a white screen from the DLP illumination

The initial intensity levels of the RGB channels in a color line CCD

The adjusted intensity levels of the RGB channels
Steep surfaces combined with gauge block (heights of 1.12 mm, 1.14 mm, and 1.15 mm)

- Sample Area: 10mm x 25mm
- Measurements were repeated 10 times and the measurement repeatability were respectively 0.34 μm and 0.24 μm.

<table>
<thead>
<tr>
<th>Step Height</th>
<th>10 μm</th>
<th>20 μm</th>
</tr>
</thead>
<tbody>
<tr>
<td>The measured average step heights</td>
<td>10.54 μm</td>
<td>20.78 μm</td>
</tr>
</tbody>
</table>

Sub-micrometer measuring accuracy and high repeatability have been achieved!

* The primary limitation arises from the camera occlusion or shadow caused by steep profile just like the traditional projection fringe method.
3D Profile: Sapphire Substrate

- 4-inch diameter
- The flatness of the substrate based on the minimum zone evaluation of surfaces is 2.33 μm.
- By comparison with 3.25 μm peak to valley value obtained from a white light interferometer at 1 nm resolution, the measurement uncertainty was found to be roughly 1 μm.

The standard deviation after five trials is \(~0.25 \text{ μm}\) which shows possible accuracy in micrometer order and high precision in the sub-micrometer scale.

It can be controlled to speeds of up to 100 mm/s thus making it possible to inspect a 4-inch substrate to within a **test duration of 1 second**.
Co-planarity of a BGA Substrate

- The measured results of co-planarity of 3.4 μm with a measurement repeatability of 0.32 μm were obtained.

Wide-field image of a BGA substrate

Sample Area: 35 mm x 35 mm

1750 line images

3D surface profile of the BGA substrate
Cloud-Based Analysis System
Why Cloud?

- Collaborations via Cloud

Reference: activeco.com

- Safely synchronize design activities via Cloud

Reference: claranet.de

Reference: accellian.com
Next-Generation Collaborations via Cloud

Controlled CCD Modules

Controlled Light Source

Parallel Image Processing

Automated Inspection

Parallel Analysis

Computing Array

Automated Optimization and Control

Management via Cloud

Hierarchical structure

Distributed structure
Comparison of Various Collaboration Models

- A multidisciplinary design optimization problem has been solved by same amount of computing nodes on Cloud but using two different collaboration models.

  - Hierarchical model
    - 10 iterations
    - 432 function evaluations
    - 30 units of working time

  - Distributed model
    - 14 iterations
    - 168 function evaluations
    - 42 units of working time
Numerous methods have been developed for surface defect detection; however, little has addressed a situation where both speed and precision requirements are satisfied simultaneously.

In our research, the requested inspection requirement is measurement area of 28 cm x 23 cm within 4 seconds with the resolution of 3.5 μm x 3.5 μm.

An expandable Distributed Image Sensor Computing System (DISCS) has been developed to achieve in-line surface defect detection.

- The hardware architecture consists of independent machines that form a master-slave parallel computing model
- The software architecture consists of MPI processes that run in CPUs and CUDA threads that run in GPUs.
Conclusion and Discussion (Continued)

• Measurement of 3D profile topography has been developed using moiré techniques.
  ▪ Straight-line grating was projected on the object surface using digital light processing (DLP) illumination.
  ▪ Tri-linear colored CCD grabbed the successive line images with 120° phase difference between each intergerogram.

• The measurement range of the proposed grating projection module and image capture module is flexible from few millimeters to hundreds of millimeters.

• The measurement speed up to 100 mm/s is possible.

• Automated optical inspection of moving substrates on a motorized transition stage has been demonstrated and is suitable for in-line or in-process inspection of conveyed products.

• The proposed method is a very good choice for non-contact profilometry because the inspection process can be handled remotely using simple instruments operating at high speed, yet providing good accuracy, high resolution, and insensitivity to environmental noise.