elektron ámbar

Original Article

Revista elektron, Vol. 6, No. 1, pp. 1-7 (2022)
https://doi.org/10.37537/rev.elektron.6.1.144.2022

Sistema de inspección de defectos en baldosas cerámicas implementado en FPGA

Ceramic Tile Defect Inspection System Implemented on FPGA

Tomás Medina [#1], Martín Vázquez [*2], Lucas Leiva [*3]

[#] *Departamento de computación y sistemas, UNICEN*
Tandil, Bs. As., Argentina
[1] tmedina@alumnos.exa.unicen.edu.ar

[*] *Laboratorio Sistemas Embebidos, INTIA, UNICEN*
Tandil, Bs. As., Argentina

Facultad de Ingeniería, Universidad Nacional de Tres de Febrero
Caseros, Buenos Aires, Argentina
{[2] mvazquez,[3] lleiva}@labset.exa.unicen.edu.ar

Recibido: 28/02/22; Aceptado: 04/05/22

***Resumen*— La modernización en las fábricas es un factor clave para la producción y la calidad del producto final. Sin embargo, esta modernización puede suponer una inversión que las empresas no pueden asumir, dejándolas fuera de la adaptación a la Industria 4.0. En la industria de fabricación de baldosas cerámicas se utilizan inspecciones visuales para determinar la calidad del producto final. Estas tareas son realizadas generalmente por operarios expuestos a entornos de riesgo. Este trabajo presenta una solución de bajo costo para la inspección automática de baldosas cerámicas. Los defectos analizados son gotas, defectos de material, esquinas, bordes y dimensiones. Todos los algoritmos se implementaron en SoC FPGA (dispositivo Xilinx Zynq) utilizando síntesis de alto nivel. Los algoritmos se verificaron y validaron en un entorno controlado construido para evaluar aplicaciones de inspección visual. Los resultados de utilización de recursos y tiempos de procesamiento indican que la implementación en una línea de producción real es factible.**

Palabras clave: Síntesis de alto nivel, FPGA, Visión Computacional, detección de gotas, detección de pinholes, inspección morfológica.

***Abstract*— Modernization in factories is the key for production and product quality. However, this modernization can involve an investment that companies cannot take, leaving them out of adaptation to Industry 4.0. In the ceramic tile manufacturing industry visual inspections are used to determine the quality of the final product. These tasks are carried out by operators exposed to risky environments. This work presents a low-cost solution for automatic ceramic tile inspection. The defects analyzed are blobs, pinholes, corners, edges and dimensions. All the algorithms were implemented on SoC FPGA (Xilinx Zynq device) using high level synthesis. The algorithms were verified and validated in a controlled environment built to evaluate visual inspection applications. The results of resource utilization and processing times indicate that the implementation in a real production line is feasible.**

Keywords: High Level Synthesis, FPGA, Computer Vision, Blob Detection, Pinhole Detection, Morphological Inspection.

I. Introducción

El control de calidad es la parte del proceso de manufactura en la que se asegura que los productos cumplan con los requerimientos mínimos definidos por la empresa, tanto en etapas intermedias o finales en la fabricación de un producto. Esto permite la verificación de su aptitud para ser comercializado y por esta razón es fundamental en cualquier proceso de fabricación [1]. Durante la fabricación de baldosas cerámicas, es necesaria una inspección visual al final del proceso para detectar si existen defectos morfológicos o en la superficie de la misma, de manera que si no cumple con los estándares de calidad del fabricante sea retirada de su comercialización. En estas industrias, la mayoría de las etapas de fabricación se encuentran automatizadas, con excepción de la etapa de inspección visual [2]. Generalmente esta tarea es realizada por operarios que se exponen a los peligros de trabajar en un ambiente hostil para la salud debido a los riesgos físicos de trabajar entre máquinas, con contaminación acústica. Además resulta una tarea repetitiva que genera fatiga tras largas jornadas y subjetiva ya que depende del juicio de clasificación del operador, y lo que un trabajador considere que es un defecto a otro puede no parecerle así. Por otra parte, existen distintos tipos de defectos que pueden estar presentes en el producto final, como por ejemplo: gotas (*blobs*), fallas de material (*pinholes*), rayones (*scratchs*), roturas (*cracks*) [3].

Existen diferentes soluciones comerciales de sistemas de inspección visual para automatizar estas tareas, como por ejemplo las ofrecidas por Cognex [4] o Stemmer Imaging [5]. Sin embargo, sus altos costos (superando los 100 mil Euros) hacen difícil su incorporación en las industrias nacionales. Como alternativa a ello, se encuentra el desarrollo de una solución *ad-hoc* mediante el uso de herramientas que permiten el prototipado rápido y la configuración o adaptación de sistemas de visión, tales como Matrox Imaging Library (MIL) y NI Vision Builder.

Además, también se encuentran disponibles bibliotecas de código abierto como OpenCV, SimpleCV, Visualization toolkit (VTK), Darwin, y ROVIS [6].

En cuanto a plataformas de implementación de sistemas industriales, se destacan los microcontroladores, los DSPs y los FPGAs. Estos últimos son propicios en estos ambientes por su alta velocidad, flexibilidad, paralelismo inherente y la gran variedad de recursos lógicos especializados disponibles [7]. Con el advenimiento de herramientas más avanzadas para FPGA, como Vivado HLS y su integración con OpenCV (a través de la biblioteca *xfOpenCV*), los desarrolladores de software pueden generar sistemas de visión por computadora de tiempo real con más facilidad [8].

En la literatura se encuentran numerosos trabajos publicados de soluciones para la inspección visual automática de baldosas cerámicas. En [9] se presenta un análisis de la literatura al respecto, describiendo y comparando los resultados de más de 200 trabajos relevantes. En el reporte indica que existen una gran cantidad de trabajos basados en observaciones estadísticas y métodos basados en filtros, como los presentados en [3].

Este trabajo presenta el desarrollo y evaluación de algoritmos de: detección de defectos por goteo de pintura, detección de defectos de material (*pinholes*), e inspección de defectos en bordes, esquinas y dimensiones. Los algoritmos son aptos para la industria de fabricación de baldosas cerámicas con texturas aleatorias, migrando las soluciones presentadas en [3] (basado en PC) a soluciones íntegramente basadas en SoC FPGA. Se pretende que sean implementadas en la línea de producción de TandilCeram (ex Loimar). Los algoritmos se describieron utilizando Vivado HLS, y se validaron sobre un kit de desarrollo Zybo Z7-20 de Digilent, que posee un SoC FPGA de Xilinx Zynq XC7Z020-1CLG400C. La captura de las imágenes se realizó mediante una Pcam 5C de Digilent, que cuenta con un sensor OV5640 e interfaz de conexión MIPI CSI-2, configurada con una resolución de 1920x1080 (1080p).

En la sección II del trabajo se presenta la descripción del entorno. En la sección III se describe el algoritmo de detección de gotas. En la sección IV se presenta el algoritmo de detección de defectos de material. En la sección V se expone el algoritmo de inspección de bordes, esquinas y dimensiones. En la sección VI se presentan los resultados experimentales obtenidos de la simulación e implementación, y finalmente en la sección VII se describen las conclusiones y trabajos futuros.

II. Plataforma base de desarrollo

Se utilizó la plataforma y metodología presentada en [10] incorporando los algoritmos de detección de defectos e inspección (módulo abstracto *VisualInspection*) dentro de la configuración, y siguiendo las recomendaciones de su uso, tal como se presenta en el diagrama de bloques de la Figura 1. El módulo IP *VisualInspection* se realizó utilizando Vivado HLS 2019.1, y Vivado 2019.1 para la integración con el resto de la plataforma.

En el sistema propuesto, el procesador (PS, por sus siglas en inglés) ejecuta el programa que permite la configuración del formato de imágenes del módulo IP que interactúa con la Pcam 5C, el factor de corrección Gamma, y la resolución y frecuencia de salida de imágenes. Además interpreta los resultados generados por el módulo *VisualInspection*.

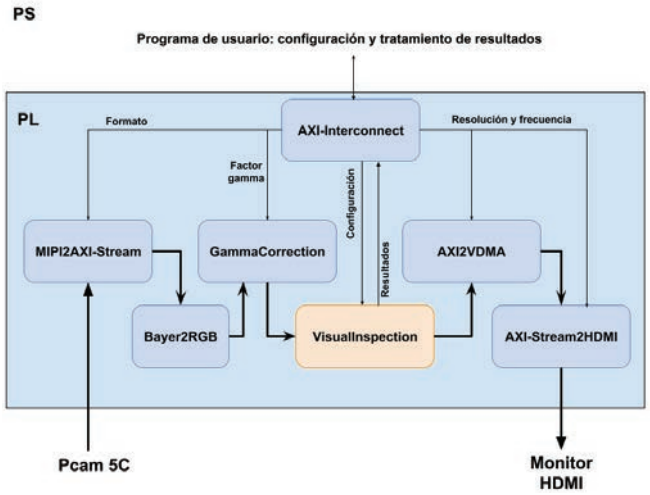

Fig. 1. Diagrama en bloques de la configuración del sistema dentro de la plataforma presentada en [10].

Por otra parte, la lógica programable (PL, por sus siglas en inglés) se configura con los IPs provistos por la plataforma, incorporando el IP de inspección visual en el flujo de procesamiento de video.

Para la detección de defectos de gotas de pintura y defectos de material, el módulo IP desarrollado en HLS, si bien permite detectar y localizar los defectos, la lógica de marcado en imágenes resultantes se excluye de la implementación final, debido a que implica un consumo excesivo en requerimientos de memoria, ya que se realiza luego de la ejecución del algoritmo. En cambio, el marcado en el algoritmo para determinar los defectos morfológicos se puede realizar concurrentemente con el procesamiento. Por tanto, los resultados sólo son transmitidos al procesador (localización de defecto y ubicación). La lógica de salida presenta una etapa previa del algoritmo. Sin embargo la lógica de marcado de resultados es utilizada para la simulación y verificación del algoritmo. Si bien el módulo actualmente funciona como un *bypass* de imagen hacia la presentación en pantalla (*AXI2VDMA*), la conexión se mantiene para preservar el pipeline de procesamiento de imagen y con capacidad futura de incorporar lógica de marcado de resultados.

En el caso de los defectos morfológicos (bordes o esquinas rotas), los resultados sí son marcados en la imagen resultante.

III. Algoritmo de detección de gotas

Las gotas (*blobs*) son un tipo de defecto que se originan por el goteo de material de tonalizado. Debido a la naturaleza de su formación, estas imperfecciones se presentan con forma circular en la superficie de la baldosa. Además, denotan un color más oscuro, como se presenta en el ejemplo de la Figura 2.

El algoritmo propuesto para la detección de presencia de gotas se compone de cuatro etapas:

- *Preprocesamiento y segmentación*: la imagen es leída y procesada con el objetivo de marcar los *blobs* candidatos. En primer lugar, se delimita la imagen según su región de interés (*ROI*, por sus siglas en inglés), el cuál es el interior de la baldosa. A continuación, debido a que su color es oscuro y presentan gran tamaño, se aplican las funciones de

xfOpenCV *xf::threshold()*, *xf::bitwise_not()*, *xf::erode()* y *xf::dilate()* para destacar este tipo de objetos.

- *Detección de objetos*: se extraen los contornos y se determina las características de los *blobs* candidatos, almacenando en un arreglo de datos la información respecto al área, perímetro y posición en la imagen. Si bien en OpenCV los contornos pueden ser extraídos utilizando la función *cv::findContours()*, esta función no está provista en xfOpenCV. En consecuencia, se analizaron algoritmos de detección de *blobs* de una pasada (a fin de minimizar la ocupación de memoria y mejorar el rendimiento) y se realizó una implementación basada en el algoritmo de detección de objetos conectados propuesto en [11].
- *Medición de características*: a cada objeto detectado se le calcula su ancho, alto, perímetro y redondez según los datos calculados en la etapa anterior. Para calcular su redondez, se utiliza la Ecuación (1).
- *Clasificación de gotas*: define si cada *blob* candidato corresponde a una gota o no según su área y redondez. Estos parámetros pueden ser dinámicamente configurados desde el programa ejecutado en el procesador.

$$redondez = \frac{área * 4 * \pi}{\sqrt{perímetro}} \quad (1)$$

Fig. 2. Imagen en detalle de dos defectos de gotas en la superficie de una baldosa.

En la Figura 3 se presenta un esquema de estas etapas y el flujo de los datos entrantes y salientes de cada una.

Para la optimización de tiempos, se utilizaron las directivas disponibles en Vivado HLS *DATAFLOW* (aporta paralelización de procesamiento a nivel de funciones), *PIPELINE* (paralelización a nivel operación en la detección de objetos y medición de características), *LOOP UNROLL* (para desenrollamiento de bucles en la clasificación de gotas). Por otro lado, para la optimización de utilización de recursos se usan los tipos de datos numéricos eficientes que ofrece Xilinx (*ap_uint*, *ap_float*), y directivas *RESOURCE* y *STREAM* para configurar los recursos a utilizar con los arreglos e imágenes respectivamente. Con el fin de minimizar los tiempos de cálculo, el perímetro de los círculos a utilizar en (1) se calcula previamente y almacena en una tabla, lo que reduce el tiempo de cálculo. Esta tabla se implementa como arreglo sobre una memoria RAM de tipo LUTRAM de dos puertos para el acceso a dos datos en simultáneo, y reducir aún más el tiempo de procesamiento.

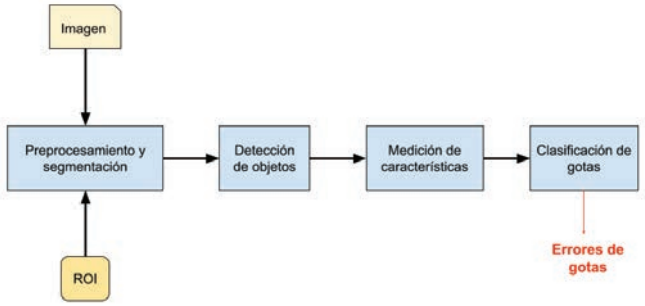

Fig. 3: Esquema de las etapas de procesamiento del algoritmo de detección de gotas.

IV. Algoritmo de detección de defectos de material

Los defectos de material (*pinholes*) son pequeños defectos en la superficie de la baldosa que se producen durante la cocción del material. Al igual que las gotas, presentan una forma circular, pero son de menor diámetro y en general más oscuros. Un ejemplo de este tipo de defectos se puede observar en la Figura 4.

Fig. 4: Imagen en detalle defecto de tipo *pinhole* en la superficie de una baldosa.

Debido a que comparten algunas características con las gotas (alto nivel de redondez y de color más oscuro), la solución implementada es similar a la detección de gotas y comparte las primeras tres etapas con el algoritmo anterior. Las etapas son:

- *Preprocesamiento y segmentación*.
- *Detección de objetos*.
- *Medición de características*.
- *Filtrado de objetos*: se hace un filtrado adicional con respecto al algoritmo anterior, con el fin de mejorar la detección y disminuir la cantidad de *pinholes* candidatos a clasificar en el paso siguiente. Esta etapa consiste en descartar los pinholes candidatos cuya diferencia entre alto y ancho sea mayor a cierto umbral.
- *Clasificación de pinholes*: define si cada *pinhole* candidato corresponde a un defecto de material o no según su área y redondez. Estos parámetros pueden ser dinámicamente configurados desde el programa ejecutado en el procesador.

En la Figura 5 se presenta un esquema de estas etapas.

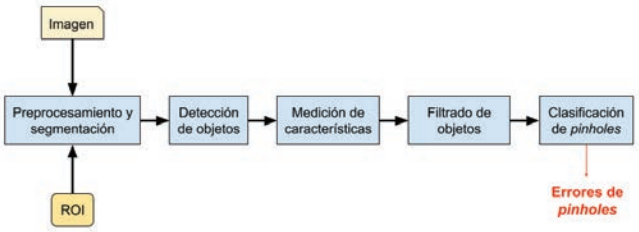

Fig. 5: Esquema de las etapas de procesamiento del algoritmo de detección de *pinholes*.

Las directivas de síntesis utilizadas para la implementación de este algoritmo son equivalentes a las empleadas en el algoritmo de detección de gotas.

V. Algoritmo de inspección de bordes, esquinas y dimensión

Las baldosas fabricadas pueden presentar defectos en su morfología. Los errores más comunes de este tipo son esquinas o bordes rotos (Figura 6) y dimensiones incorrectas (baldosa con área por debajo o por encima de los valores de calidad adecuados).

Fig, 6: Imagen en detalle defectos en bordes (izquierda) y esquinas (derecha).

El algoritmo propuesto se basa en la segmentación de la imagen para obtener la silueta de la baldosa, y con ella cuantificar los defectos morfológicos. El algoritmo comprende las siguientes etapas:

- *Preprocesamiento y segmentación*: la imagen es leída y procesada con el objetivo de marcar la silueta de la baldosa. Para ello se utilizan las funciones de xfOpenCV *xf::rgb2gray()*, *xf::threshold()*, *xf::erode()* y *xf::dilate()*.
- *Detección de corners*: se utiliza el algoritmo de Harris-Stephen Corner Detection [12] para la detección de esquinas, mediante la función de xfOpenCV *xf::cornerHarris()*.
- *Clasificación de corners*: se clasifican los *corners* detectados, identificando las esquinas de las baldosas (delimitación de la región de interés o ROI). Además, realiza una corrección de la distorsión del lente, debido a que la cámara utilizada posee distorsión radial de tipo *barrel effect*. En este paso se detectan los defectos de esquinas y bordes rotos.
- *Cálculo del área*: según la posición de las esquinas, se calculan el alto, ancho y área de la baldosa. Con ello se determina si tiene las dimensiones correctas.

En la Figura 7 se presenta un esquema de las etapas y los datos que arroja cada una.

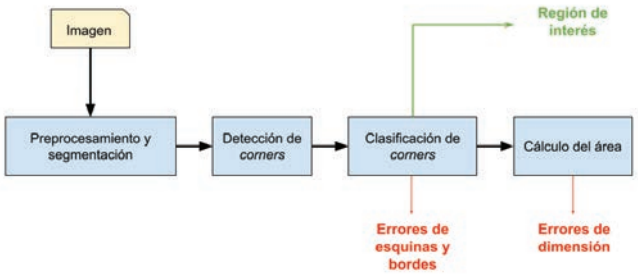

Fig. 7: Esquema de las etapas de procesamiento del algoritmo de detección de inspección de bordes, esquinas y dimensiones de la baldosa.

Se utilizaron las directivas provistas en Vivado HLS *DATAFLOW* y *PIPELINE* para la paralelización de las distintas etapas y la directiva *STREAM* para la inferencia de las imágenes como flujos de datos. Además, como en los demás algoritmos, se utilizan los tipos de datos para una síntesis eficiente en hardware.

VI. Resultados experimentales

El algoritmo fue validado utilizando un ambiente controlado construido a medida. El entorno de captura se compone de una caja de interior negro, con una bandeja desplazable a través de guías en la cual se apoyan las baldosas a analizar. La caja posee una iluminación led con intensidad regulable, y la entrada se encuentra cubierta por una cortina para evitar el ingreso de luz exterior. En la parte superior del ambiente se encuentra ubicada la cámara la cual permite la captura cenital de los elementos a inspeccionar. En la Figura 8 se puede ver una fotografía del exterior de este ambiente de pruebas.

Fig. 8: Ambiente de pruebas utilizado para tomar fotografías de las baldosas.

Utilizando el ambiente y la configuración provista en [10] (Pcam 5C y Zybo Z7-20) para adquisición de imágenes, se realizó la captura de imágenes de un lote de 36 baldosas cerámicas para la verificación funcional de los algoritmos de inspección en alto nivel. En la Fig. 9, se

presentan dos imágenes de ejemplo correspondientes al banco de imágenes de muestras obtenido.

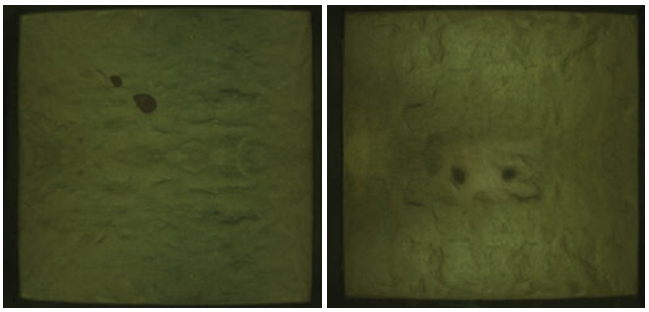

Fig. 9: Imágenes capturadas con Zybo Z7-20 y Pcam 5C de baldosas con defectos de gotas utilizando la configuración de captura de [10].

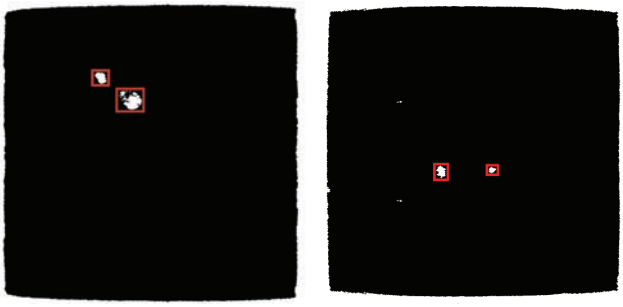

Fig. 10: Imágenes resultantes de la detección de gotas sobre baldosas con defectos generadas por la simulación en Vivado HLS.

Se realizaron pruebas de simulación en software (Vivado HLS) sobre las imágenes obtenidas, resultando en una detección efectiva de presencia de gotas de pintura en las baldosas. La Figura 10 presenta el resultado de la simulación para las imágenes de la Figura 9, en donde se marcan en rojo la ubicación de las gotas detectadas por el algoritmo. Asimismo, se verificó con el conjunto de imágenes completo arrojando resultados negativos en la detección en todas las imágenes que no contenían este tipo de defectos, y el subconjunto total de muestras con presencia de gotas fue reconocido como tal, dando una precisión del 100%.

El módulo IP fue sintetizado y se obtuvieron los resultados de utilización de recursos que se muestran en la Tabla I. Se puede observar que la ocupación es relativamente baja en comparación a los recursos disponibles en el FPGA de la Zybo Z7-20, siendo el recurso más utilizado las LUTs con un 21% del total.

TABLA I
RESULTADOS DE UTILIZACIÓN DE RECURSOS

Recursos	*BRAM_18K*	*DSP48E*	*FF*	*LUT*
Total	30	3	5150	11583
Disponible	280	220	106400	53200
% Utilización	10%	1%	4%	21%

En cuanto a tiempos estimados, suponiendo que se analizan 32 objetos candidatos a ser defectos de gotas, se obtuvo que la latencia mínima y máxima son de ~125.000 y ~51.000.000 ciclos de reloj respectivamente. Esto puede variar si existen más objetos a medir y clasificar, pero se toman 32 según la cantidad máxima de objetos en las imágenes de prueba. Tomando una frecuencia de reloj de 150 MHz, con el caso de la latencia máxima se puede procesar una imagen cada 0,255 segundos, o casi 4 imágenes por segundo.

Para la verificación del segundo algoritmo (de detección de *pinholes*) se utilizó el mismo conjunto de 36 imágenes de prueba, de las cuales 8 poseen estos defectos de material. En la Figura 11 se presenta el detalle de un pequeño defecto de este tipo.

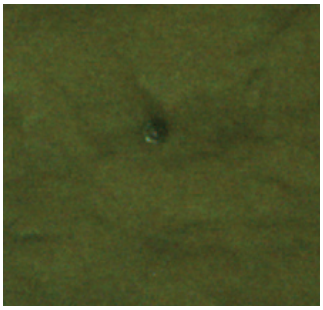

Fig. 11: Detalle de defecto de material en imagen capturada con Zybo Z7-20 y Pcam 5C.

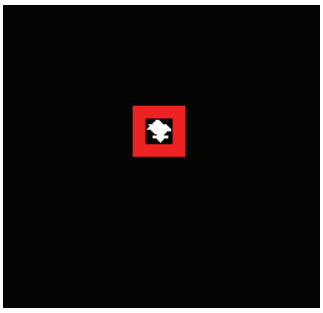

Fig. 12: Detalle de defecto de material en imagen capturada con Zybo Z7-20 y Pcam 5C.

En las pruebas de simulación en software sobre las imágenes obtenidas la detección de defectos fue exitosa, logrando identificar las 8 baldosas con defectos. La Figura 12 presenta el resultado de la simulación para la imagen de la Figura 11, en donde se marca en rojo la ubicación del *pinhole* detectado. Además, en el banco de imágenes completo no se evidenciaron falsos negativos. El subconjunto total de muestras con presencia de *pinholes* fue reconocido como tal, dando una precisión del 100%.

Se realizó la síntesis del módulo IP obteniendo los resultados de ocupación de recursos que se muestran en la Tabla II. Se puede observar que si bien comparte muchas características, etapas y funciones de biblioteca utilizadas en el algoritmo de detección de gotas, la utilización de recursos aumenta, sobre todo en LUTs. Esto se debe a la incorporación de la etapa adicional de filtrado de objetos. A pesar de esto, los recursos disponibles en la FPGA de la Zybo Z7-20 siguen siendo suficientes para su implementación.

TABLA II
RESULTADOS DE UTILIZACIÓN DE RECURSOS (PINHOLES)

Recursos	*BRAM_18K*	*DSP48E*	*FF*	*LUT*
Total	31	6	12678	22799
Disponible	280	220	106400	53200
% Utilización	11%	2%	11%	42%

Respecto al tiempo de ejecución, suponiendo que se analizan 32 objetos candidatos a ser defectos de *pinholes*, al igual que en el algoritmo de detección de gotas se obtuvo que la latencia máxima y mínima son de ~125.000 y ~51.000.000 ciclos de reloj respectivamente. De igual manera que en el algoritmo anterior, esto varía si existen más objetos a medir y clasificar. Con una frecuencia de reloj de 150 MHz, con el caso de la latencia máxima se puede procesar una imagen cada 0,255 segundos.

En cuanto al tercer algoritmo, se realizó la verificación del algoritmo de inspección de bordes, esquinas y dimensión sobre el banco de imágenes de 36 baldosas con defectos, de las cuales 4 tienen defectos de bordes, 4 defectos de esquinas, y 4 de dimensión del tipo más pequeñas de lo aceptado. En la Figura 13, en la imagen de la izquierda se ve el detalle de un ejemplo de una baldosa con una esquina rota, y por la derecha un borde defectuoso.

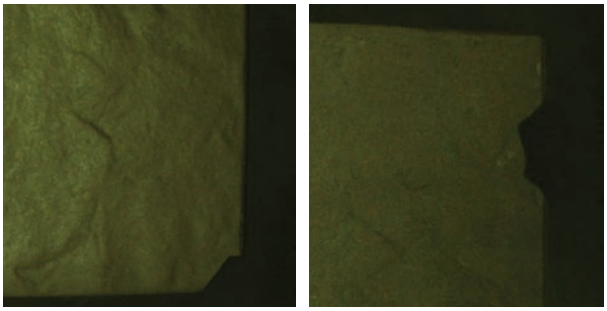

Fig. 13: Detalle de defecto de esquina (izquierda) y de borde (derecha) en imagen capturada con Zybo Z7-20 y Pcam 5C.

La inspección fue exitosa en los casos de prueba, ya que se logró detectar los defectos correspondientes en cada imagen del banco de imágenes. En la Figura 14 se pueden observar los resultados de la detección de defectos de las baldosas de la Figura 13, resaltando en rojo las partes detectadas como defectuosas. Por otro lado, no se detectaron errores en las demás imágenes que contenían errores de otro tipo, por lo que con este algoritmo también se tiene una precisión del 100%.

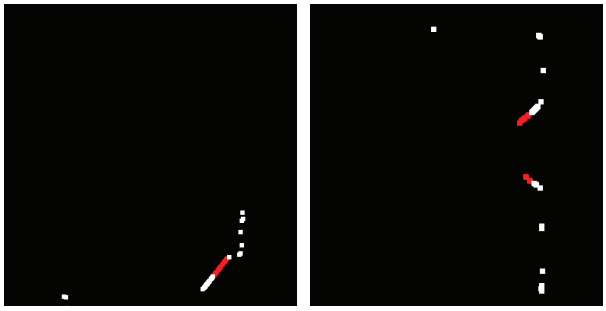

Fig. 14: Resultados de la detección de defectos de esquinas (izquierda) y bordes (derecha).

Tras verificar la precisión del algoritmo, se realizó la síntesis del módulo IP, la cual arroja los resultados de utilización de recursos que se muestran en la Tabla III. Se observa que el tipo de recurso más utilizado son las Look-Up Tables, utilizando el 37% del total disponible en el chip. Esta alta utilización es mayormente provocada por la síntesis de la función de biblioteca que realiza la detección de *corners*, (Harris-Stephens).

TABLA III
RESULTADOS DE UTILIZACIÓN DE RECURSOS (INSPECCIÓN DE BORDES, ESQUINAS Y DIMENSIÓN)

Recursos	***BRAM_18K***	***DSP48E***	***FF***	***LUT***
Total	61	13	11027	19288
Disponible	280	220	106400	53200
% Utilización	21%	5%	10%	37%

El algoritmo desarrollado tiene una latencia tanto mínima como máxima de ~4.000.000 de ciclos de reloj. Tomando una frecuencia de reloj de 150 MHz, una imagen puede ser procesada en 0,027 segundos (37 imágenes por segundo).

Por último, tras verificar el correcto funcionamiento y que los tiempos estimados de cada algoritmo se adaptan a los requerimientos, se realizó la co-simulación en Vivado HLS con éxito, verificando que el resultado del diseño sintetizado a nivel RTL se condice con la simulación en software de cada uno de ellos.

La integración de los tres algoritmos en un único dispositivo es posible considerando que las implementaciones reutilicen funciones compartidas, principalmente las etapas de los algoritmos de detección de gotas y *pinholes*, y que la información visual de los resultados no sea transmitida por video. De esta forma, la ocupación lógica de la integración de los algoritmos involucra el uso de 36417 LUTs, 22498 FFs, 16 DSP48Es, y 91 BRAM_18Ks, que incorporados a la plataforma presentada en [10] da una ocupación final de 99,4% de LUTs, 43,1% de FFs, 7,2% de DSP48Es y 52,5% de BRAM_18Ks.

La solución propuesta presenta mejoras de rendimiento respecto a [3], que detalla la solución basada en una PC con un procesador Intel Core i7-7700, 16 GB de memoria RAM y disco SSD, y requiere 0,4 segundos para la ejecución de los tres algoritmos implementados en este trabajo (0,34 segundos para la detección de defectos de bordes, esquinas y dimensión, 0,03 segundos para la detección de gotas y 0,03 segundos para la detección de *pinholes*). En cambio, utilizando SoC FPGA es posible explotar el paralelismo a través del hardware, y ejecutar los algoritmos concurrentemente en un tiempo total de 0,255 segundos.

VII. CONCLUSIONES

El trabajo describe la implementación y validación de un sistema de inspección visual basado en FPGA que permite detectar defectos de presencia de gotas de pintura, defectos de material y defectos en bordes, esquinas y dimensiones en una línea de producción de baldosas cerámicas. Los resultados obtenidos del prototipo experimental desarrollado cumplen con los requerimientos esperados, demostrando la factibilidad de incorporar un sistema complejo a menor costo en ambientes industriales. Esto posibilita la incorporación de tecnología eficiente a bajo costo en las industrias nacionales.

También, se pudo observar que el desarrollo *ad-hoc* de la solución permite incorporar funcionalidades a medida del cliente, permitiendo incluso el monitoreo remoto o sistema IIoT (*Industrial Internet of Things*). En este aspecto, es posible aprovechar las diferentes conexiones con las que

cuenta el kit, para transmitir los resultados a través del puerto Ethernet, por ejemplo.

Se resalta que no siempre es eficiente aplicar la metodología provista por Xilinx, que propone partir de una implementación solo en OpenCV. La biblioteca xfOpenCV contiene sólo un subconjunto de funciones de OpenCV, y las funciones no soportadas incrementan considerablemente el tiempo de desarrollo.

Como trabajos futuros se pretende continuar con los algoritmos restantes de detección de fallas (*cracks*, *scratches*), y realizar los ensayos experimentales en la línea de producción real con el desarrollo del prototipo operacional. Además se propone el tratamiento de datos resultantes a fin de adaptar el sistema a la Industria 4.0, principalmente en el área de IIoT. Además se continuará con optimizaciones de los algoritmos, como por ejemplo en la minimización de recursos del algoritmo de detección de esquinas, a través del uso de métodos alternativos, como por ejemplo *FAST Corner Detection*.

Agradecimientos

Este trabajo fue parcialmente financiado por la SeCAT de UNICEN (Código de Proyecto 03/C287).

Referencias

[1] D. T. Pham, R. J. Alcock, Smart Inspection Systems: Techniques and applications of intelligent vision, Academic Press, 2003.

[2] G.M.A. Rahaman, M. Hossain, “Automatic Defect Detection and Classification Technique from Image: A Special Case Using Ceramic”, Int. J. Comput. Sci Inf. Secur., vol. 1, n 9, 2009.

[3] L. Echeverz, M. Matías Melograno, L. Leiva. "Inspección automática de defectos de superficie en baldosas cerámicas." XXIV Congreso Argentino de Ciencias de la Computación, La Plata, 2018.

[4] Cognex. (n.d.). In-Sight 7000 Series Vision Systems. Retrieved may 11, 2021, from https://www.cognex.com/products/machine-vision/2d-machine-vision-systems/in-sight-7000-series

[5] Stemmer Imaging. (n.d.). Machine vision systems. Retrieved may 20, 2021, from https://www.stemmer-imaging.com/en/products/category/vision-systems/

[6] Z. Liu, H. Ukida, H., P. Ramuhalli, K. Niel. Integrated Imaging and Vision Techniques for Industrial Inspection. Springer, 2015.

[7] J.J.R. Andina, E. De la Torre Arnanz, M.D. Valdes. FPGAs: Fundamentals, Advanced Features, and Applications in Industrial Electronics. CRC Press, 2017.

[8] A. Swirski. "TULIPP and ClickCV: How the Future Demands of Computer Vision Can Be Met Using FPGAs." Towards Ubiquitous Low-power Image Processing Platforms. Springer, Cham, 2021, pp. 235-259.

[9] T. Czimmermann, G. Ciuti, M. Milazzo, M. Chiurazzi, S. Roccella, C. M. Oddo, P. Dario. "Visual-based defect detection and classification approaches for industrial applications—a survey." Sensors 20, no. 5 (2020): 1459.

[10] T. Medina, L. Leiva, M. Vázquez. "Plataforma para Procesamiento de Imágenes sobre SoC FPGA de Xilinx." Elektron vol. 4, no. 2, 2020, pp. 81-86.

[11] J. Trein, A. Th Schwarzbacher, B. Hoppe. "FPGA implementation of a single pass real-time blob analysis using run length encoding." MPC-Workshop, February, 2008.

[12] C. Harris and M. Stephens, “A Combined Corner and Edge Detector,” The 4th Alvey Vision Conference, Manchester, 31 August-2 September 1988, pp. 147-151.