Universidad Tecnológica Nacional
AGOSTO 2021 / Año 19- Nº 41

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.41.75-90.2021 - ISSN 1666-6933

Algoritmo de detección e Interfaz gráfica de usuario diseñados en Python para autocolimador digital.

Detection algorithm and GUI designed in Python for digital autocollimator.

Presentación: 04/03/2021

Aprobación: 12/07/2021

Guillermo J. Bergues
Centro de Investigación y Transferencia en Metrología (CEMETRO), Universidad Tecnológica Nacional – Argentina
gjbergues@gmail.com

Clemar Schürrer
Centro de Investigación y Transferencia en Metrología (CEMETRO), Universidad Tecnológica Nacional – Argentina
cschurrer@gmail.com

Nancy Brambilla
Centro de Investigación y Transferencia en Metrología (CEMETRO), Universidad Tecnológica Nacional – Argentina
nancybrambilla@gmail.com

Resumen

En este trabajo se presenta una interfaz gráfica de usuario y se explica el modelo de detección de líneas diseñado en Python para la misma. Luego de los resultados de la calibración del instrumento electrónico se procedió a crear la interfaz gráfica de usuario en python en base al prototipo de la GUI construida en Matlab para lo cual se tuvo que reescribir el algoritmo de detección en el mismo lenguaje. El instrumento electrónico está construido con un autocolimador analógico comercial, Nikon 6D, y una interfaz visual ad hoc.

Palabras Claves: autocolimador electrónico; python; detección de líneas, interfaz visual, GUI.

Abstract

In this work, the graphical user interface is presented and the line detection model designed in Python for it is explained. After the results of the calibration of the electronic instrument, the graphical user interface in python was created based on the prototype of the GUI built in Matlab, for which the detection algorithm had to be written in the same language. The electronic instrument is built with a commercial analog autocollimator, Nikon 6D, and

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.41.75-90.2021 - ISSN 1666-6933

an ad hoc visual interface.

Keywords: electronic autocollimator; Python; line detection, visual interface, GUI.

Introducción

Un autocolimador es un instrumento óptico para medir ángulos sin contacto. Se puede usar, entre otras mediciones, para definir la rugosidad de una superficie con el método Moody (Burton, 1997). La medición en el Autocolimador Nikon 6D es realizada por un operario mediante el posicionamiento de uno de sus ojos en el ocular del instrumento. Este tipo de medición no solo introduce errores del tipo aleatorio y sistemático debido a la presencia de paralaje, propio del método utilizado, sino que también expone al operario a un continuo cansancio visual. Tanto para reducir las incertidumbres de medición y a la vez, mejorar la experiencia de medición para el operario, se suelen crear interfaces visuales para los instrumentos ópticos (Yuan, 2005; Alcock et al., 2010; Tan et al., 2007). Este ha sido el caso del autocolimador Nikon 6D que a través de sucesivos desarrollos se convirtió en un instrumento digital. La resolución del instrumento sin interfaz es de 0,5 segundos de arco, mientras que con la digitalización se logra una R=0,012 segundos de arco, que da como resultado un aumento de 41 veces en la capacidad de medida. A continuación, se detalla tal proceso que permitió esta mejora.

Las etapas del desarrollo y diseño fueron las siguientes: en primer lugar, se realizó un modelo digital de la retícula del instrumento para obtener la relación píxel-segundo de arco (Bergues et al., 2013), luego se creó el prototipo de medición con una cámara de baja resolución y sensor CMOS (Schürrer et al., 2014) y en base a ella se implementó una interfaz de alta resolución (Bergues et al., 2014). Una vez montada esta última, se estudiaron e implementaron diferentes procesamientos de imágenes para obtener el algoritmo óptimo de detección (Bergues et al., 2015).

Con el algoritmo que mide la posición de las líneas en base al cálculo del máximo de las superficies con cresta que poseen los segmentos se estudió la deformación del sistema óptico y su desalineación. En el experimento montado para este análisis se midieron los desplazamientos del plano (x; y) de la cámara + Lente con un conjunto de 3 palpadores de alta resolución Mahr-Federal LVDT. Estos palpadores pueden medir en un rango de ±250mm con una resolución de 0,1µm. El tamaño del píxel de la cámara utilizada es 4,4µm, por lo cual estos palpadores pueden medir desplazamientos a nivel sub-píxel (1/44 de píxel). Como la resolución de los palpadores es muy alta, estos se apoyaron sobre superficies de vidrio con rugosidad menor a 10nm. El sistema traslacional se movió en un rango máximo de ±0,15mm, tanto vertical como horizontalmente, armando una grilla de captura mecánica para la cámara. Los pasos regulares se generaron manteniendo la cámara en foco y asegurando que el campo visual de la misma no capture el borde del ocular (cuestión que introduce sombras indeseadas en las imágenes). Se armó una matriz de captura conformada por 120 puntos, que corresponden a 120 imágenes de la escala. Todas estas imágenes fueron procesadas con el algoritmo de detección sub-pixel (Bergues et al., 2017).

Este estudio permitió obtener el cálculo de incertidumbre del nuevo instrumento (Bergues et al., 2018). Finalmente, se realizó la calibración del dispositivo con el Patrón Nacional de Ángulo del Instituto Nacional de Metrología de Argentina (INTI) y se estudió la performance del instrumento a partir de los resultados de la calibración (Bergues et al., 2020).

Estos avances sustanciales propiciaron el paso a seguir para la funcionalidad y operatividad

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.41.75-90.2021 - ISSN 1666-6933

del instrumento. De esta manera, la etapa siguiente, tal cual se propone en otros estudios y en diversos campos (Ismail et al., 2013; Du et al., 2015; Tara et al., 2017; Phutthanukun et al., 2019) es la programación de la interfaz visual de usuario con mayor detalle. Esta interfaz tiene el objetivo de agregar confiabilidad a la vez que otorga, con sus características, un continuo avance de las funcionalidades del instrumento.

En busca de este objetivo, en primer lugar, se desarrolló el prototipo de la interfaz gráfica de usuario en base a Matlab, debido a que este entorno de programación y diseño permitía combinar rápidamente el cálculo matemático y el procesamiento de imágenes necesario para la interfaz visual.

En este trabajo se presenta el diseño de la interfaz visual en Python a la vez que se explican los algoritmos y librerías necesarias para este objetivo. Es necesario traducir todos los algoritmos escritos en Matlab a Python que, si bien requiere de una programación matemática más compleja, ofrece varias ventajas a futuro. Las razones de la elección de Python son, entre otras, que es un lenguaje de programación abierto, multiplataforma, dinámico, flexible y con buena legibilidad.

En cuanto al desarrollo del presente trabajo, en la Sección 2 se explica el concepto general de la interfaz y el instrumento óptico mejorado (Autocolimador Nikon 6D). A la vez se desarrollará el algoritmo de detección en su concepción inicial. En la Sección 3 se explica en detalle el algoritmo en base al código Python. En la Sección 4 se muestran las dos interfaces gráficas de usuario, tanto la desarrollada en Matlab como en Python y se comenta su funcionalidad. Finalmente, en la Sección 5, están las conclusiones del diseño.

Algoritmo de detección.

1. Autocolimador e interfaz visual

En la Fig. 1 se observa el autocolimador Nikon 6D junto con su interfaz electrónica en frente del ocular del mismo. La interfaz es la encargada de solucionar los temas mencionados con anterioridad y está constituida por una cámara marca Basler con sensor CCD, un esqueleto de aluminio que sostiene la cámara y una PC donde está instalado el algoritmo de procesamiento.

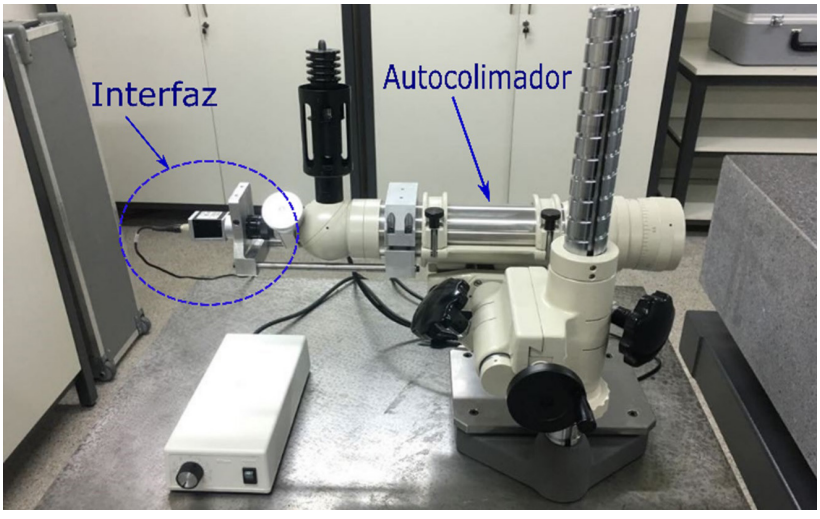


Fig. 1. Autocolimador Nikon 6D junto a su interfaz visual (cámara Basler), ubicada frente al ocular del instrumento.

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.41.75-90.2021 - ISSN 1666-6933

2 Medición de superficies.

Una de las mediciones usualmente realizadas con autocolimadores es la obtención de la topología de una superficie, experiencia esquematizada en la Fig. 2, donde se ve sobre la mesa de referencia (superficie) la barra de aluminio que sirve de guía al espejo E para su desplazarlo en línea recta. Con esta guía se mueve el espejo E y se toman varias mediciones de las inclinaciones angulares de la superficie en esa zona. Luego estas inclinaciones son traducidas en elevaciones de la superficie; de esta manera, se obtiene la rectitud de la superficie en la dirección de desplazamiento. La práctica repetida en las 8 direcciones que conforman el rectángulo (el plano del granito) con sus diagonales y medianas permiten mediante el método Moody obtener la topología de la superficie plana en estudio.

La Fig. 3 muestra la gráfica sencilla, utilizada a modo de explicación, de la medición implementada en la Fig. 2. La función g(x) representa la altura de cada punto de la superficie a lo largo de la dirección de desplazamiento del espejo sobre la superficie a medir, llamada x. Para cada posición del espejo E definimos como Nx a la normal a la superficie, en el punto x. Si llamamos N0 a la recta inicial de referencia (en x=0) entonces:

$$\frac{dg(x)}{dx} = tg\left(\alpha_{M(x)}\right) \tag{1}$$

La topología de la superficie en la dirección X, definida como la rectitud de X en el plano X-Z en metrología dimensional se obtiene a partir de los ángulos (α_M) medidos en radianes convertidos en elevaciones en micrómetros por el paso adaptado, que está definido por el largo del espejo E y el ángulo de inclinación (Burton, R. ;1997):

$$g(x) = g(0) + \int_0^x tg\left(\alpha_M\right).dx \tag{2}$$

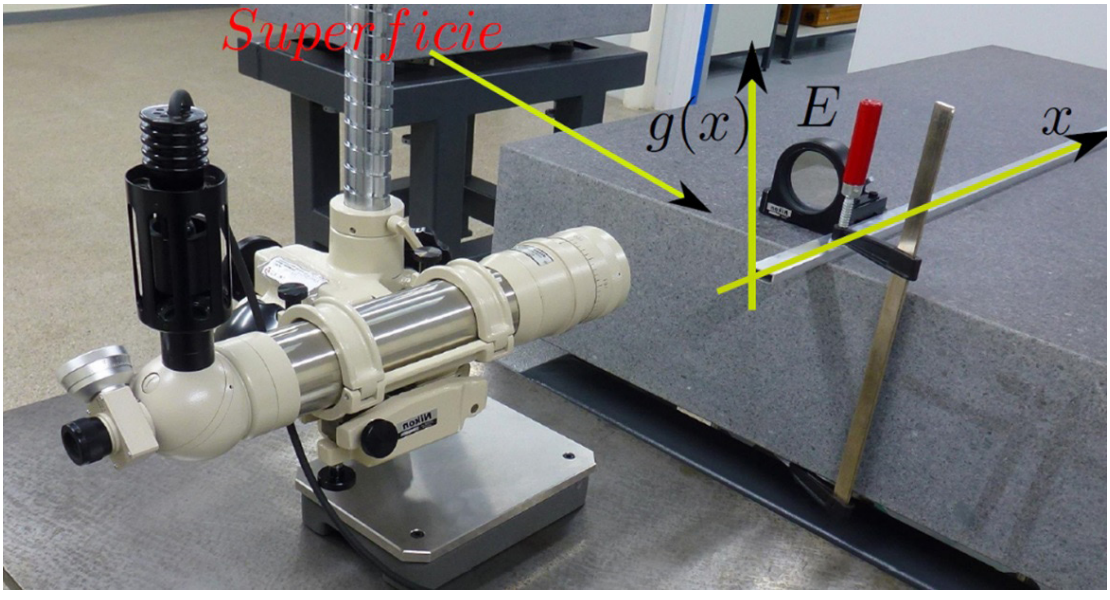


Fig. 2. Método para la medición de la topología de una superficie de granito con autocolimador. El espejo E se utiliza para captar la forma de la superficie.

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.41.75-90.2021 - ISSN 1666-6933

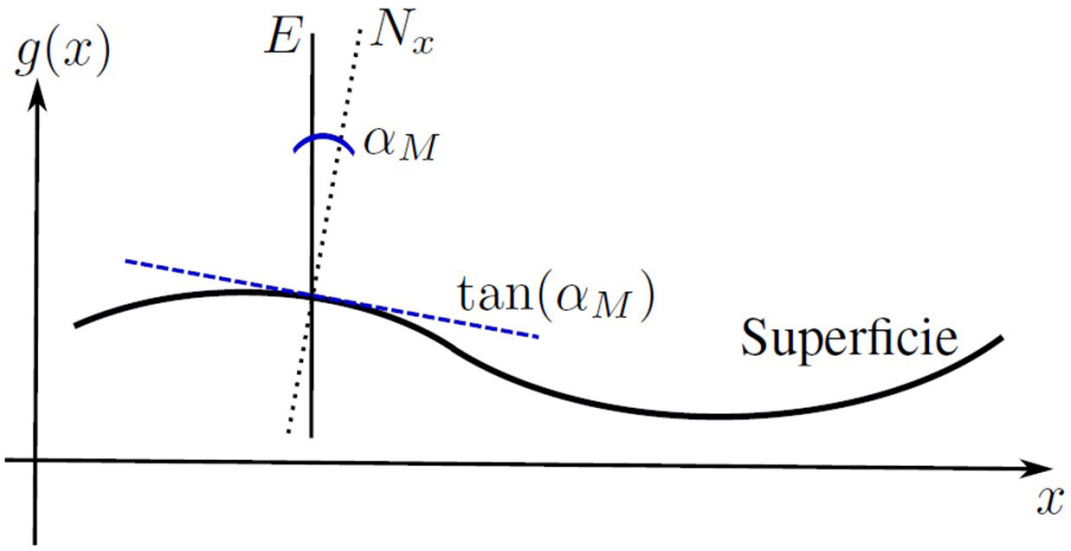


Fig. 3. Esquema indicando el sistema de coordenadas y los parámetros utilizados en la medición.

3. Escala y Cruz.

En la Fig. 4 podemos ver un esquema de la retícula interna del autocolimador, en donde se distinguen los elementos con los cuales se discierne una medición, la escala marcada en minutos y el haz de luz en forma de cruz reflejado por el espejo. Observando la posición de la cruz con respecto al centro de la retícula se obtiene la medición de los ángulos de cabeceo y guiñada entre la superficie y el espejo E (Schürrer et al., 2014).

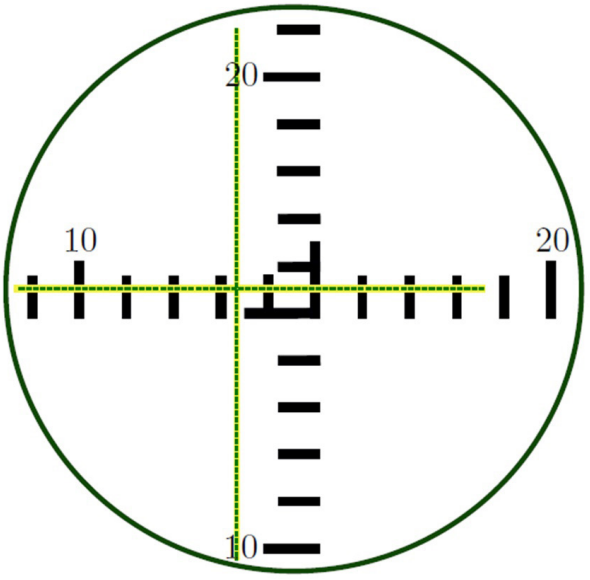


Fig. 4. Esquema de la retícula, donde se observa la escala interna del autocolimador y el haz de luz en forma de cruz.

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.41.75-90.2021 - ISSN 1666-6933

Para realizar la medición en forma automática de los valores esquematizados en la Fig. 4 se deben seguir los pasos del algoritmo que se especifican a continuación:

1. Establecer la distancia Δ_{XY} entre las divisiones de la escala de la retícula a nivel sub-píxel con estimación de incertidumbre. En la Fig. 5 se identifica este valor tanto para los segmentos horizontales como verticales, marcando su medición desde los centros de los segmentos.
2. Identificar las líneas que forman la cruz, también a nivel sub-píxel y,
3. establecer la distancia B_x y B_y en píxeles entre el cero de la retícula (el valor central de la misma donde los segmentos forman el ángulo recto, ver Fig. 5), y el corte de las líneas que forman la cruz con cada eje.

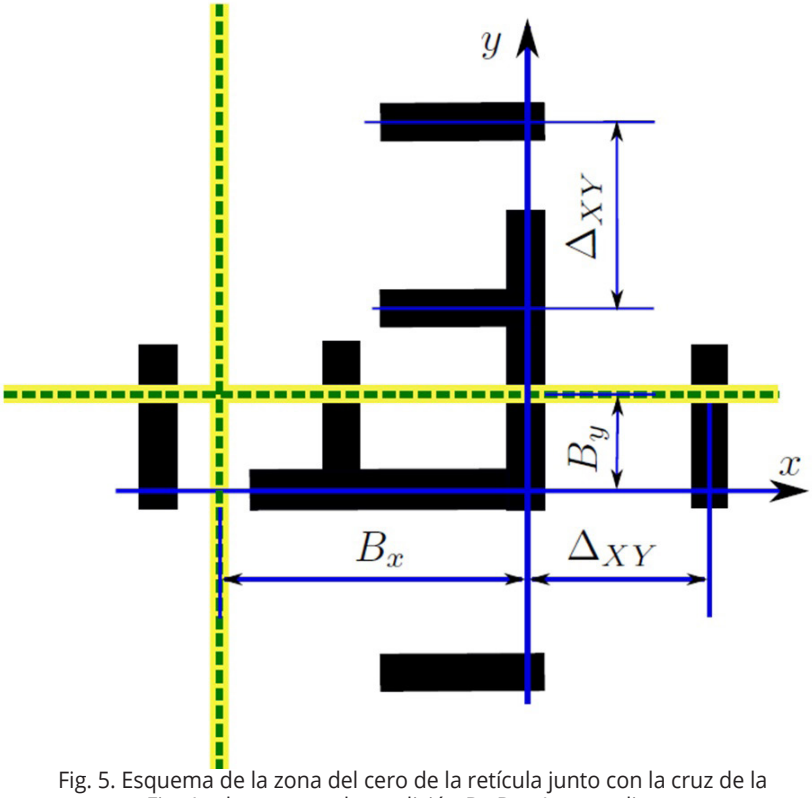


Fig. 5. Esquema de la zona del cero de la retícula junto con la cruz de la Fig. 4, y los puntos de medición B_x, B_y y Δ_{XY} a analizar.

En el autocolimador Nikon 6B/6D (el procedimiento sirve para ambos modelos B y D), la distancia entre divisiones de la retícula (Δ_{XY}) representa 60 segundos de arco, (1div = 1min). Por lo cual para obtener la medición del ángulo de cabeceo αy en segundos de arco, se calculan primero los valores Δ_{XY} en píxeles/división y B_y en píxeles, y se reemplazan dichos valores en (2):

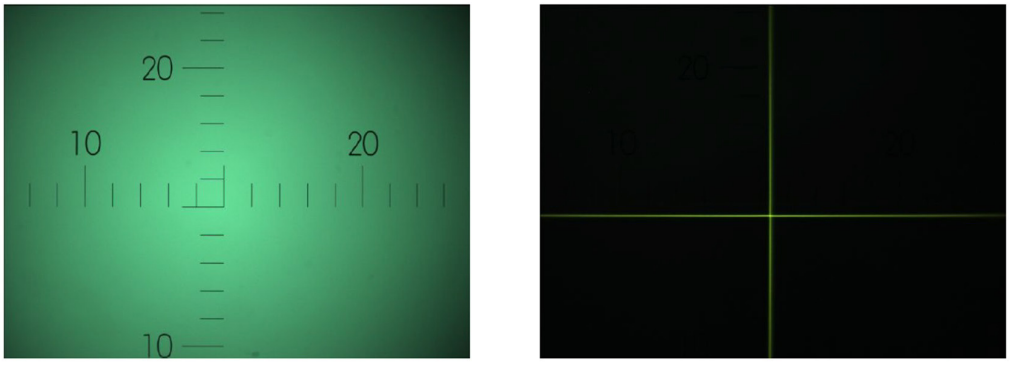
Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.41.75-90.2021 - ISSN 1666-6933

$$\alpha_y = 60\frac{B_y}{\Delta_{XY}} \tag{3}$$

Lo mismo se puede hacer para el ángulo de guiñada, αx, en segundos de arco.

4. Método de procesamiento: ajuste gaussiano

La interfaz digital genera dos grupos de imágenes color, un grupo corresponde a la escala de la retícula del autocolimador (Fig. 6a), y otro al del haz de luz en forma de cruz (Fig. 6b), que aparecen juntas como se ve en el esquema de la Fig. 4. Ambos grupos de imágenes digitales contienen figuras conformadas por conjuntos de líneas rectas cuya posición tiene que ser determinada en forma muy precisa a nivel sub-píxel para inferir la posición correcta de la línea en la grilla de medición.



(a) Retícula de Medición. (b) Haz de luz en forma de cruz.

Fig. 6. Imágenes capturadas con la cámara CCD de la interfaz.

Para trabajar estas imágenes se utiliza el método de ajuste de funciones continuas del tipo gaussiano para ajustar las muestras de la curva que surgen del corte unidimensional perpendicular a la línea de la imagen (ver Fig. 7, Fig. 8 y Fig. 9). La posición de la línea sub-píxel se encuentra al calcular el valor pico de la función. Los métodos de ajuste requieren un modelo paramétrico de línea para ser aplicados, de la misma manera que lo hace la transformada Hough. Entonces para desarrollar el algoritmo a partir de la imagen capturada en primer lugar se deben obtener cada uno de los valores de máximo gradiente que determinan la posición de la línea. En la dirección dada por el gradiente se realiza un corte de la línea y se ajusta un modelo gaussiano en ese corte con parámetros que representen la posición sub-píxel de la línea (ver Fig. 7).

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.41.75-90.2021 - ISSN 1666-6933

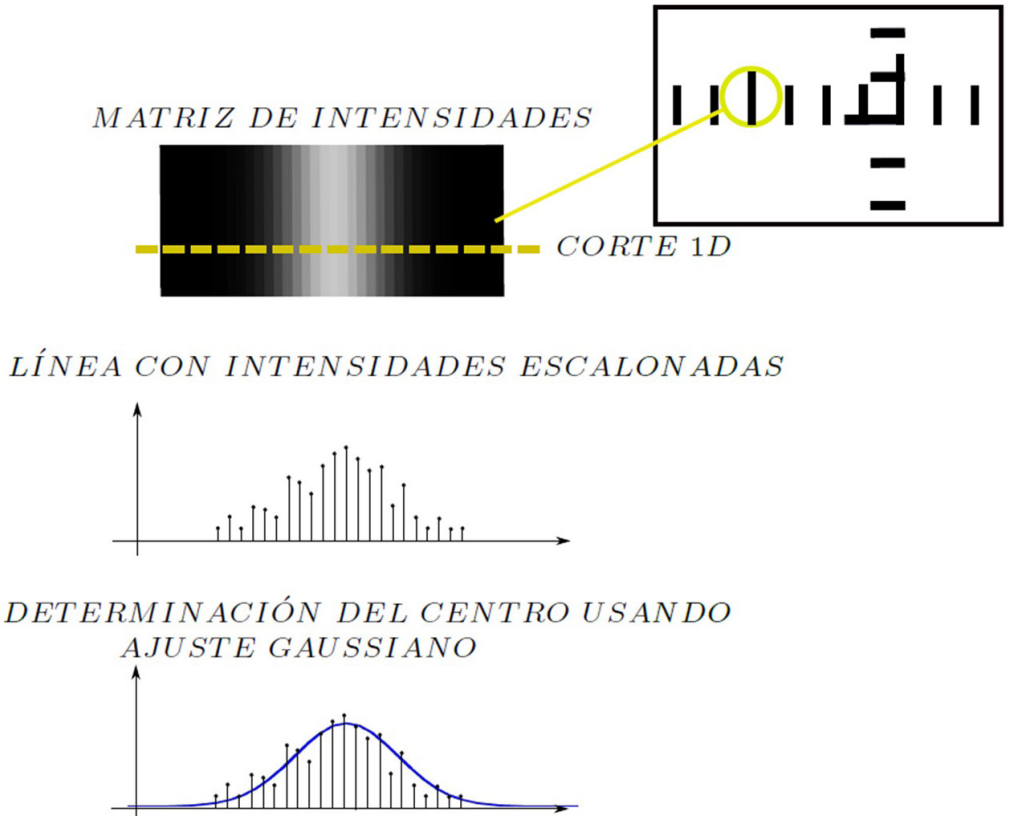


Fig. 7. Algoritmo de procesamiento de los segmentos y líneas de la escala y cruz.

Algoritmos en Python.

La escala y la cruz de la Fig. 6 tienen variantes del algoritmo de la Fig. 7. En este apartado se explican las diferentes ramificaciones a la vez que se va desarrollando todo el programa en sus funciones principales. Todo el código está disponible en un repositorio de github (https://github.com/gjbergues/IV_python.git).

1. Conexión de la cámara.

La conexión se logró mediante una coordinación entre las librerías openCV, PIL y pypylon. Esta última es una nueva librería creada por Basler para el manejo de sus productos a través de Python. Es de código abierto, por lo cual se la puede adecuar a las diferentes aplicaciones. A continuación, se presentan las líneas principales del código propuesto:

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.41.75-90.2021 - ISSN 1666-6933

```
cam.StartGrabbing()
# converting to OpenCV bgr format
converter = pylon.ImageFormatConverter()
converter.OutputPixelFormat = pylon.PixelType_BGR8packed
converter.OutputBitAlignment = py-lon.OutputBitAlignment_MsbAligned

if cam.IsGrabbing():
    # Wait for an image and then retrieve it.
    grabResult = cam.RetrieveResult(5000, py-lon.TimeoutHandling_
ThrowException)
    if grabResult.GrabSucceeded():
        image = converter.Convert(grabResult)
        img = image.GetArray()
    grabResult.Release()
cam.StopGrabbing()
```

Los valores mostrados en este código corresponden en su mayoría a las características de la librería pypylon, salvo la función de conversión. Para la posterior etapa, luego de la captura, se utilizan las funciones de OpenCV, que son las encargadas de la manipulación de la imagen, por esto mismo se remarca esta parte del código: es necesaria la conversión de formatos de imagen. Esto se debe hacer para que la imagen pueda mostrarse en el objeto Tkinter.

2. Escala: Efecto *vignetting*.

La escala sufre del efecto vignetting debido a la incidencia de los rayos luminosos en el telescopio del autocolimador. Este efecto se relaciona a una pérdida de iluminación en la salida de un sistema óptico, principalmente debido al bloqueo de una parte del haz de rayos incidente por el tamaño efectivo del tope de apertura, lo que resulta en un desvanecimiento gradual de la imagen en puntos cercanos a su periferia (Hecht, 2001). Dado el carácter continuo y parabólico del fondo que se forma en la imagen debido a esta deformación, el filtro Savitzky-Golay es un método ideal para reducir el efecto. Este filtro está diseñado para suavizar imágenes (Schafer, 2011) aumentando la relación señal-ruido (S/N) sin distorsionar mucho la señal. Esto se logra ajustando sucesivos subconjuntos de puntos de datos adyacentes con un polinomio de segundo grado utilizando el método de mínimos cuadrados. Para realizar este procedimiento en Python es necesario usar la librería scipy.optimize para crear un plano de regresión a los datos de la escala. Para lo cual se sigue este procedimiento:

1. Colocar la imagen en 3 vectores columna.
2. Hacer una regresión lineal –linalg.lstsq
3. Evaluar en la grilla de los vectores unidimensionales.

3. Escala: Cálculo de la distancia Δ_{XY}.

Una vez que el efecto vignetting es reducido se procede a obtener el centro de la escala. Esto se puede realizar obteniendo el valor de máxima intensidad que se corresponde naturalmente, en este caso, con el centro de la imagen. A continuación, se indica un ejemplo del cálculo para la posición vertical:

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.41.75-90.2021 - ISSN 1666-6933

```
for j in np.arange(start=100, stop=1500, step=1):
 cv = img[:, j] # vertical cuts
 s = np.sum(cv)
 if s > sm:
 ind_c = j
 sm = s
```

Luego, se obtiene un corte de la imagen para hacer un ajuste gaussiano como se muestra en la Fig. 7. En este caso es necesario crear una ventana que recorra cada uno de los segmentos y a la vez obtenga su posición tal como se muestra en la Fig. 8.

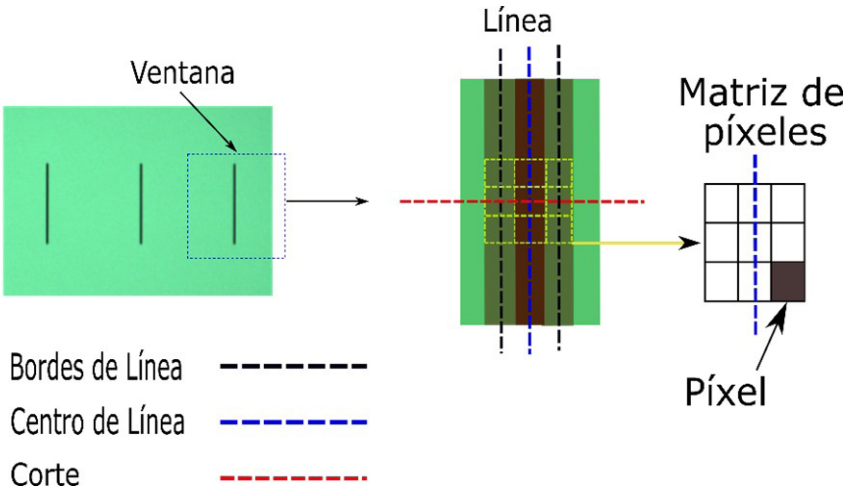


Fig. 8. Ventana de análisis para obtener centro de línea. Se tomaron todos los segmentos de la escala, corriendo la ventana de izquierda a derecha.

Para el ajuste gaussiano se utiliza la rutina curve_fit de la librería scipyp.optimize:

```
mean = sum(x * y) / sum(y)
sigma = np.sqrt(sum(y * (x - mean) ** 2) / sum(y))
# pco_v = estimated covariance of pop_t.
pop_t, pco_v = curve_fit(gauss, x, y, p0=[min(y), max(y),mean, sigma])
# This is the gaussian function "gauss":
H + A * np.exp(-(x - x0) ** 2 / (2 * sigma ** 2))
```

4. Cruz: centro grueso para posicionamiento inicial.

El centro de la cruz se tiene que obtener de manera diferente porque la misma recorre toda la imagen a diferencia de la escala, donde los segmentos son más pequeños. Luego de realizar el proceso inicial de alineación que describe (Bergues et al., 2014), se pueden hacer cortes secuenciales en varios puntos de la cruz como muestra la Fig. 9.

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.41.75-90.2021 - ISSN 1666-6933

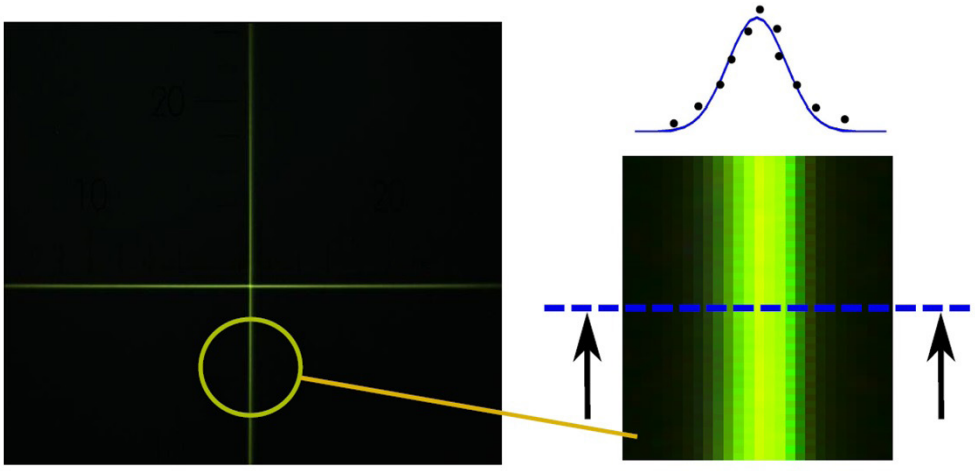


Fig. 9. Cortes secuenciales de la cruz de medición para obtener el centro de la misma.

El código para aplicar el procedimiento de la Fig. 9 es el siguiente:

```
# Y center position
ind_list_v = []
for j in np.arange(start=1, stop=1624, step=1):
   cv = img[:, j] # vertical cuts
   # Get the indices of maximum element in numpy array
   ind_v = np.argmax(cv)
   ind_list_v.__iadd__([ind_v])
ind_r = np.mean(ind_list_v)
```

Se obtiene la media del conjunto de cortes para calcular la aproximación al centro de línea a nivel píxel. No hace falta obtener el centro a nivel subpíxel porque ese es otro procedimiento que se realiza posteriormente. En esta primera etapa solo se busca la posición gruesa en la imagen.

5. Cruz: centro sub-píxel de medición.

Dado que la cruz se desplaza por toda el área de medición, es necesario definir el procedimiento de acuerdo a la posición en la pantalla, teniendo en cuenta la deformación y a los defectos de la cámara (Bergues et al., 2017). De esta manera se recomienda seguir el siguiente lineamiento:

1. Obtener posición de referencia del centro respecto al rango del instrumento.
2. Calcular centro sub-píxel.
3. Aplicar correcciones (ver ejemplo de la curva de calibración para eje X, Fig. 10).

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.41.75-90.2021 - ISSN 1666-6933

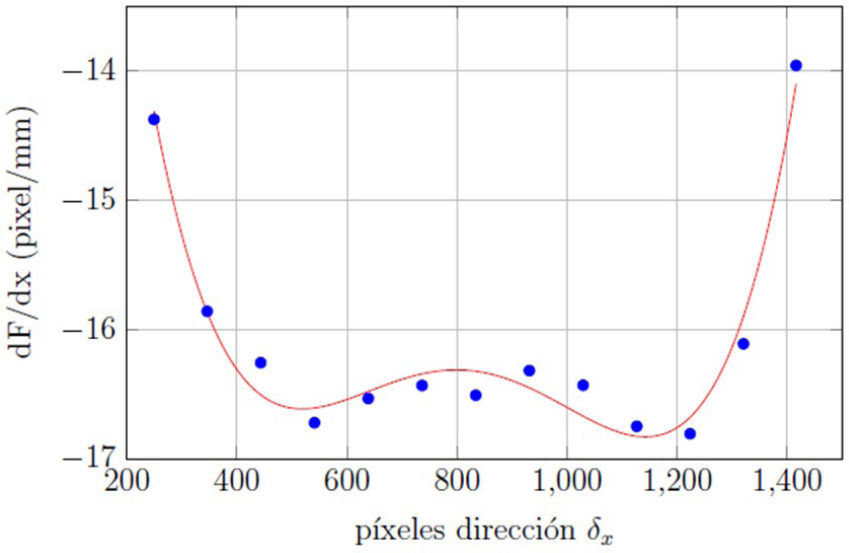


Fig. 10. Curva de calibración para la dirección x. Derivada de la función de desplazamiento en la dirección "x".

La curva de calibración surge del análisis detallado de las deformaciones de la cámara. En el caso del ejemplo de la Fig. 10, es para la dirección "x".

GUI Python.

En la Fig. 11 se puede ver la GUI construida mediante Matlab. A medida que se avanzó en la investigación se encontraron limitantes con esta GUI: es un software propietario de código cerrado y no es una opción óptima para codificar un software complejo por la dificultad en la escalabilidad. La barrera principal se encuentra en la distribución del producto final, ya que esta plataforma requiere de una licencia paga, de alto costo, para hacerlo.

Por otro lado, Python es un lenguaje de programación gratuito y de código abierto. No solo se puede descargar Python sin costo, sino que también se puede modificar el código fuente. Además, dado que Python está disponible sin costo y con una distribución de software de fácil instalación, una audiencia mucho más amplia puede usar el código desarrollado.

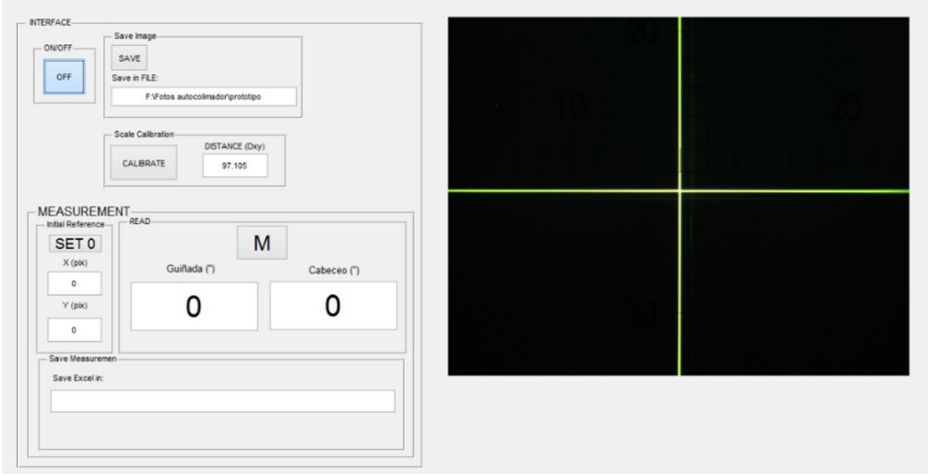


Fig. 11. Interfaz de usuario del autocolimador electrónico Nikon 6D (Matlab).

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.41.75-90.2021 - ISSN 1666-6933

La programación de la GUI con Python se desarrolló con la librería tkinter, que tiene varias opciones de construcción para diseñar botones y ventanas de información para el usuario:

```
from tkinter import *

# GUI Main Window
root = Tk()
root.title('Visual Interface')
```

En la Fig. 12 se puede ver el resultado del diseño. Que, si bien se basó en el prototipo, posee muchas funcionalidades más para el usuario, en especial para ayudarlo durante el proceso de medición.

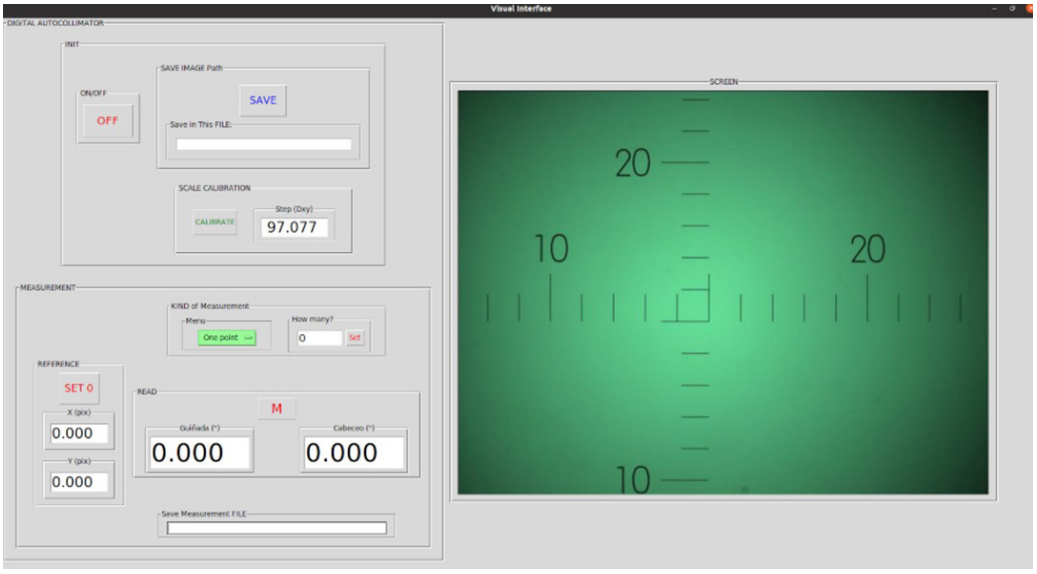


Fig. 12. Interfaz de usuario del autocolimador electrónico Nikon 6D (Python).

Dentro de las funcionalidades posee:

1. Guardar imágenes: tanto de la escala como de la cruz en el path que desee el usuario.
2. Calibra la escala y otorga nuevo valor de Δ_{XY} para los diferentes experimentos o mediciones.
3. Se creó un menú para definir una medición de punto único (para detección de ruidos) o una medición de varios puntos.
4. Posee un display de referencia para que se pueda ubicar el punto inicial de la medición donde el operario decida (a nivel píxel).
5. El display de guiñada y cabeceo es de mayor tamaña para visualizar correctamente los datos de cada punto.
6. La medición se guarda en formato .xlsx en el path que decida usuario.

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.41.75-90.2021 - ISSN 1666-6933

Conclusiones.

Este artículo propone el diseño de una plataforma de procesamiento de imágenes digitales y la creación de una interfaz visual de usuario en el lenguaje de programación Python. La usabilidad del instrumento es un atributo más de la calidad del mismo, tanto de su software como de su hardware, e impacta en la confiabilidad de un instrumento, además de la exactitud y la precisión del mismo. De esta manera, el trabajo propuesto se centra en aumentar la calidad del instrumento a través del diseño del software.

El documento se concentra en la construcción de los diferentes algoritmos de procesamiento de imagen y se muestra la interfaz gráfica de usuario (GUI) usando Python denominando sus funcionalidades. El objetivo principal para el desarrollo de la GUI es establecer una conexión entre el operario y el instrumento sin necesidad de que este posicione sus ojos sobre el ocular del autocolimador a la vez que pueda obtener otros beneficios de la digitalización del instrumento como la mejora en la capacidad de medición y el procesamiento de la medición.

El sistema propuesto beneficia al operario, de tal manera, que el mismo pueda gestionar la medición desde varios aspectos: el sistema proporcionará información de comparación, análisis de datos y registro. Además, está la posibilidad de construir nuevos métodos de medición ya que la escalabilidad del Python lo permite, por lo cual, se podrá incrementar productividad, calidad y seguridad en el uso del instrumento.

Los principios básicos de las normas ISO respecto a usabilidad implican tomar en cuenta la interacción efectiva y eficaz del usuario con el producto, la facilidad de aprendizaje y de uso, la flexibilidad y robustez. Según la ISO/IEC 9126 (Botella et al., 2004), las características para evaluar la calidad de un software son: funcionalidad, confiabilidad, usabilidad, eficiencia, mantenimiento y portabilidad. Pensando en estas características, se utilizó Python, ya que desde un comienzo las otorga.

Con la interfaz Python se pueden avanzar en nuevos desarrollos para el instrumento que se forma en conjunto con el autocolimador y la digitalización. Se plantea, como trabajo futuro, preparar al sistema para que mida automáticamente mediante el método Moody que se utiliza para caracterizar superficies tales como los granitos de referencia (patrones de mármol), máquinas de medir coordenadas (MMC) y máquina herramientas (CNC). Todas estas en base a los requisitos de la norma DIN879: Surface Accuraccy.

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.41.75-90.2021 - ISSN 1666-6933

Referencias

Burton, R. N. (1997). Data analysis for surface plate calibration. Report. United States. UNT Dig. Library.

Yuan, J., Long X., y Yang, K. (2005). Temperature-controlled autocollimator with ultrahigh angular measuring precision. Review of Scientific Instruments, vol. 76, no. 12.

Alcock, S. G., Sawhney, K. J., Scott, S., Pedersen, U., Walton, R., Siewert, F., Zeschke, T., Senf, F., Noll, T., y Lammert, H. (2010). The diamondnom: A non-contact proler capable of characterizing optical figure error with sub-nanometre repeatability. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, vol. 616, no. 2-3.

Tan, J., Ao, L., Cui, J., y Kang, W. (2007). Further improvement of edge location accuracy of charge-coupled-device laser autocollimators using orthogonal fourier-mellin moments. Optical Engineering, vol. 46, no. 5.

Bergues, G., Ames, G., Canali, L., Schurrer, C. y Flesia, A. (2013). Modelado de la retícula de medición de un autocolimador Nikon 6b/6d mediante transformada houg. XV Reunión de Trabajo en Procesamiento de la Información y Control (RPIC).

Schürrer, C., Flesia, A., Bergues, G., Ames, G., y Canali, L. (2014). Interfaz visual para un autocolimador Nikon 6d mediante procesamiento de imágenes con precisión sub-píxel: un caso de estudio. Revista Iberoamericana de Automática e Informática Industrial - RIAI, vol. 11, no.3, pp. 327-336.

Bergues, G., Ames, G., Canali, L., Schurrer, C. y Flesia, A. (2014). External visual interface for a nikon 6d autocollimator. Instrumentation and Measurement Technology Conference (I2MTC) Proceedings, IEEE International, pp. 35–39.

Bergues, G. J., Canali, L., Schurrer, C., y Flesia, A. (2015). Electronic interface with vignetting effect reduction for a Nikon 6b/6d autocollimator. Instrumentation and Measurement, IEEE Transactions on, no. 99.

Bergues, G. J., Schurrer, C., Brambilla, N. y Canali, L. (2017). Misalignment contribution to the autocollimator's scale distortion. IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Turin, pp. 1-5.

Bergues, G. J., Schürrer, C. y Brambilla, N. (2018). Uncertainty Determination of the Set Nikon 6B Autocollimator + Visual Interface. IEEE Transactions on Instrumentation and Measurement, vol. 67, no. 5.

Bergues, G., Schürrer, C. y Brambilla, N. (2020). Estudio de la Capacidad de Medición y Calibración (CMC) de un Autocolimador Electrónico Nikon. Revista Tecnología y Ciencia, n. 38, p. 113-126.

Ismail, R. y Ismail, I. (2013). Development of graphical user interface (GUI) for livestock management system. 2013 IEEE 4th Control and System Graduate Research Colloquium, Shah Alam, pp. 43-47.

Du, Y., Gai, L., Tian, J., y Liu, W. (2015). Digital Image Processing Teaching Auxiliary System Based on MATLAB Graphical User Interface. 2015 7th International Conference on Information Technology in Medicine and Education (ITME), Huangshan, pp. 434-438.

Revista Tecnología y Ciencia
DOI: https://doi.org/10.33414/rtyc.41.75-90.2021 - ISSN 1666-6933

Tara, K., Sarkar, A. K., Khan, M. A. G. y Mou, J. R. (2017). Detection of cardiac disorder using MATLAB based graphical user interface (GUI). 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), Dhaka, pp. 440-443.

Phutthanukun, W. y Chayratsami, P. (2019). Development of Angle Control System Application Using Python. 2019 IEEE 11th International Conference on Engineering Education (ICEED), Kanazawa, Japan, pp. 128-132.

Hecht, E. (2001). Optics, 4th ed. Reading, MA, USA: Addison-Wesley.

Schafer, R. W. (2011). What is a Savitzky–Golay filter? [Lecture Notes]. IEEE Trans. Signal Process., vol. 28, no. 4, pp. 111–117.

Botella, P., Burgués, .X, Carvallo, J., Franch, X., Grau, G., Marco, J., Quer, C. (2004). ISO/IEC 9126 in practice: what do we need to know?