Extracción del volumen de una micropipeta con visión artificial

Julieta Karina Cruz Vázquez[1], Hugo Alejandro Jiménez García[2] & Francisco Gumaro Ruiz Ruiz[1]*

Resumen

Las micropipetas son empleadas en la manipulación de diversas muestras líquidas de origen químico y biológico, entre las que destacan, ácidos, alcoholes, muestras virales, bacterianas y fúngicas. *La manipulación de las muestras conlleva un cierto grado de riesgo a la salud del operario, en algunos casos se recomienda al usuario el menor contacto con las muestras, la solución es utilizar sistemas automatizados para su manejo, pero esta tecnología aún no es accesible en la mayoría de laboratorios o universidades por su costo elevado. Por lo anterior, se desarrolló un algoritmo de visión artificial en lenguaje de programación Python ejecutado en una microcomputadora Raspberry Pi modelo 3B, con el objetivo de la extraer la información del volumen actual de una micropipeta a través de una fotografía tomada por la cámara Raspberry Pi. El programa fue capaz de tomar la fotografía, procesar la imagen, extraer la información y generar la lectura del volumen de la micropipeta. Con esta etapa concluida, facilitará el diseño y construcción de un sistema automatizado dedicado a la manipulación de una micropipeta.*

Palabras clave: Laboratorio, plantilla, procesamiento de imagen, *raspberry pi*, robótica.

Abstract

Micropipettes are used in the manipulation of various liquid samples of chemical and biological origin, among which acids, alcohols, viral, bacterial and fungal samples. *The manipulation of the samples carries a certain degree of risk to the health of the operator, in some cases the user is recommended to have less contact with the samples, the solution is to use automated systems for their handling, but this technology is not yet accessible in the most laboratories or universities due to its high cost. Therefore, an artificial vision algorithm was developed in Python programming language executed on a Raspberry Pi model 3B microcomputer, with the aim of extracting the information of the current volume of a micropipette through a photograph taken by the Raspberry Pi camera. The program was able to take the photograph, process the image, extract the information and generate the reading of the volume of the micropipette. With this stage completed, it will facilitate the design and construction of an automated system dedicated to handling a micropipette.*

Key words: Image processing, laboratory, *raspberry pi*, robotic, template.

Recibido: 04 de septiembre de 2020.

Aceptado: 06 de abril de 2021.

[1] *Instituto de Genética, Universidad del Mar campus Puerto Escondido. Ciudad Universitaria, Vía Sola de Vega km 1.5 Carretera Puerto Escondido- Oaxaca. San Pedro Mixtepec-Juquila, Oaxaca, México, 71980.*

[2] *Estudiante de Ingeniería Mecatrónica Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas (UPIITA) del Instituto Politécnico Nacional. Ciudad de México.*

* **Autor de correspondencia:** *ruiz_ruiz_francisco@hotmail.com (FGRR)*

Introducción

Las micropipetas son dispositivos para recolectar y transferir pequeños volúmenes de líquidos (Mettler Toledo 2014); para su correcto manejo, se deben de realizar un determinado número de pasos consecutivos (Eppendorf 2020), su empleo permite el desarrollo de múltiples técnicas científicas enfocadas en áreas de la salud e investigación (Paulitschke & Nash 1993, Lee & Liu 2014). En tareas en donde se exija pipetear un gran número de muestras su uso se vuelve cansado e inclusive se convierte en una actividad imposible de realizar. Por lo anterior, algunas empresas en su sitio oficial como Eppendorf® y Mettler Toledo® ofrecen micropipetas multicanal y electrónicas, otros promueven varios modelos de pipeteadores automáticos y semiautomáticos (epMotion®96 2020), inclusive se han diseñado brazos robóticos con visión artificial para realizar las actividades de pipeteo en menor tiempo y con mayor precisión (Andrew Alliance 2019, Assist Plus 2020), pero estas soluciones automatizadas son muy costosas y muy pocos laboratorios tienen acceso a esta tecnología de primer nivel (Fig. 1).

La visión artificial es un conjunto de técnicas de análisis y procesamiento de imágenes (León *et al*. 2020), que tiene como objetivo replicar la obtención de datos que realizan los seres humanos por medio de la vista (Ortiz *et al*. 2020), numerosas aplicaciones se han documentado usando visión artificial; en la agricultura, detectando parámetros de calidad en frutas y verduras (Cubero *et al*. 2011); en la industria, en clasificación de fresas (Constante *et al*. 2016), en la industria peletera (Adamo *et al*. 2006), en sistemas de inspección de aeronaves piloteadas remotamente (Bautista *et al*. 2019), en el área de la robótica, apoyando al personal de trabajo (Bergamini *et al.* 2020), en el monitoreo de tráfico vehicular (Vergis *et al*. 2020); en el área de salud, relacionada en la identificación temprana de enfermedades crónicas (Márquez 2020) e inclusive actualmente se ha hecho presente la visión artificial ante el problema sanitario que actualmente estamos padeciendo producido por el COVID-19 (Shi *et al*. 2020).

El sistema de visión artificial se puede dividir en cuatro etapas: (1) Obtención de la imagen; en esta fase, influyen aspectos como las características del sensor ocupado, las lentes y la iluminación. (2) Pre-procesamiento: consiste en el mejoramiento de la imagen captada, aplicando una serie de filtros (cambio de color, escala de grises). (3) Etapa de filtrado o segmentación: consiste en limitar o extrae las zonas de interés para el sistema. (4) Etapa de extracción de información: al obtener la imagen de datos es necesario realizar

Figura 1. Dispositivos de pipeteo. (A) Micropipeta monocanal de volumen variable. (B) Micropipeta electrónica multicanal de volumen variable. (C) Pipeteador semiautomático de 96 puntas Epmotion®. (D) Estación de trabajo Opentrons®

algoritmos de extracción de la información pertinente (González *et al*. 2006), siendo la coincidencia de plantillas "*Template Matching*" una de las técnicas relacionadas en el procesamiento de imágenes por computadora (Cuevas *et al*. 2017).

La técnica coincidencia de plantillas "*Template Matching*" es el procedimiento más básico de detección de correspondencias entre imágenes, se han realizado en diversas aplicaciones como el reconocimiento biométrico (Valencia *et al*. 2014), en la verificación del parentesco (Goyal & Meenpal 2020), en general, el reconocimiento se realiza mediante la coincidencia de imágenes "*matching*", o regiones de imagen, estableciendo una comparación con las plantillas almacenadas representativas "*templates*" (Brunelli 2009).

Las capacidades que presenta el ser humano al realizar actividades donde se le exija una buena visión, coordinación, habilidad y concentración, pueden verse comprometidas al momento de realizar la misma actividad muchas veces en poco tiempo, la técnica de pipeteo no es la excepción, al manejar líquidos que en algunos casos son peligrosos, demanda todas esas cualidades. Por otro lado, la tecnología está en continuo desarrollo, siendo el caso de *Raspberry Pi* 3, una microcomputadora que actualmente está despertando el interés en diversos ámbitos del conocimiento. La placa *Raspberry Pi* está diseñada en miniatura, principalmente para ejecutar un sistema operativo de código abierto (Upton & Halfacree 2016), con su aplicación se han realizado aplicaciones robustas e integrales (Donat 2014), con estas ventajas que ofrece *Raspberry Pi* 3, se procedió a realizar el diseño del primer algoritmo de visión artificial orientado a la lectura del volumen de una micropipeta. Con esta etapa concluida, se espera que en un futuro sea aprovechada como punto de partida para el desarrollo de un sistema de apoyo automatizado en el manejo de micropipetas.

Material y métodos

Los materiales utilizados para el desarrollo del presente trabajo se pueden consultar en la tabla I.

La documentación fue descargada del sitio oficial de *Raspberry Pi* (Raspberry Pi 2019), se descargó la versión del sistema operativo *Raspbian Jessy* con ambiente gráfico de la página, posteriormente fue copiado en la tarjeta SD, después la tarjeta se insertó en la *Raspberry Pi* 3, inmediatamente se conectó la pantalla, el teclado y el mouse a la *Raspberry Pi* 3. Una vez encendida la *Raspberry Pi* 3 se configuró

Tabla I. Materiales para el desarrollo de la identificación del volumen con visión artificial.

Componente	Características
Rasbperry Pi Modelo 3B+	Procesador de cuatro núcleos a 1.4 GHz
Pantalla oficial de Raspberry Pi	Táctil de 7 pulgadas
Cámara para Raspberry Pi	5MP con montura M12x05
Lente	Lente telescópico
Tarjeta SD	8 GB de capacidad
Fuente de iluminación	Módulo de 16 LED (4x4) de 5volts (V)
Dos soportes universales	Metal
Dos pinzas de 3 dedos	Acero inoxidable
Teclado y mouse	Con conexión USB
Micropipetas de laboratorio	Volumen de 100 microlitros (µL)

abriendo la terminal presionando las teclas Ctrl+Alt+t. Posteriormente se introdujo el siguiente comando en la terminal "*sudo raspi-config*" (Fig. 2).

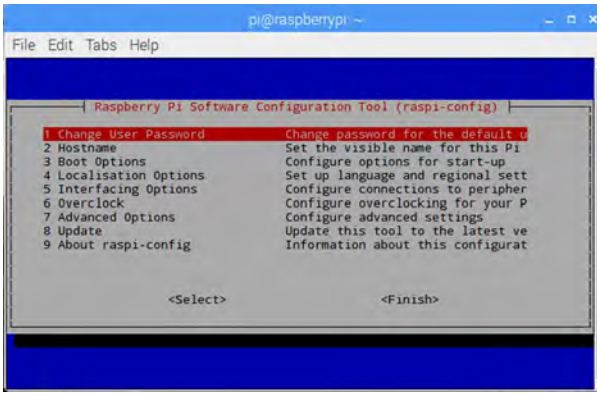

Figura 2. Configuración de la consola de *Raspberry Pi* 3.

Se seleccionó la opción "expand file system" esto permitió al sistema operativo utilizar todo el espacio en la tarjeta SD, luego de esto la *Raspberry Pi* 3 se reinició. Una vez reiniciada, nuevamente se abrió la terminal y se introdujo el comando "*sudo raspi-config*"; a continuación, se optó por la opción "*Interfacing options*" y posteriormente "*camera*" por último se presionó la tecla "aceptar".

Primeramente, se realizó una búsqueda de actualizaciones disponibles, se accedió a la terminal y se introdujo los siguientes comandos: "*sudo apt-get update*" y "*sudo apt-get upgrade*". Al haber concluido las actualizaciones, se instalaron los paquetes necesarios para realizar la identificación del volumen, con el uso de los siguientes comandos "*sudo apt-get install python*", "*sudo apt-get install python-picamera*", "*sudo apt-get install python-numpy*", "*sudo pip install opencv-python*". Después de haber terminado de instalar los paquetes, la *Raspberry Pi* 3 quedó habilitada para la identificación de los números de la micropipeta.

En base a la figura 3, para la detección de los números en el indicador de la micropipeta de 100 µL. se corroboró que la micropipeta posee una zona rectangular de 17 milímetros (mm) x 5 mm que corresponde al indicador de volumen con números arábigos impresos que denotan el volumen a la cual está preparada para medir una muestra líquida. La mayoría de las micropipetas convencionales tienen localizado el indicador del volumen en un costado del mango para su manipulación.

Los dígitos del indicador de la micropipeta se extrajeron en el siguiente orden: número superior: centenas, número central: decenas, número inferior: unidades

Una vez obtenido el valor en una variable de tipo entero "*int*", se realizaron multiplicaciones por potencias de 100, 101, 102, etc., para ajustar la lectura a la micropipeta utilizada. Con esa esa estrategia se propuso un algoritmo de visión artificial descrito en el diagrama de flujo de la figura 4.

El diagrama de flujo se implementó mediante el *script* de *Python* "*Tmatching.py*". Los templetes utilizados se realizaron para trabajar con tamaños de imagen: 1024x768 y 128x96. Las imágenes requeridas para las plantillas o templates numéricos se obtuvieron fotografiando la micropipeta mediante el *script* de *Python* "*imagetest.py*".

Las áreas de interés se recortaron manualmente y se aplicó una binarización por medio del script "*bin.py*". En el caso del template para ubicar el indicador, la imagen se recortó de una fotografía de la micropipeta, posteriormente se convirtió a escala de grises y se aplicó un filtro *Gaussian blur vertical* y horizontal usando el Programa de manipulación de imágenes GNU (Gimp 2019).

Las imágenes obtenidas para una resolución de 128x96 se presentan en la figura 5.

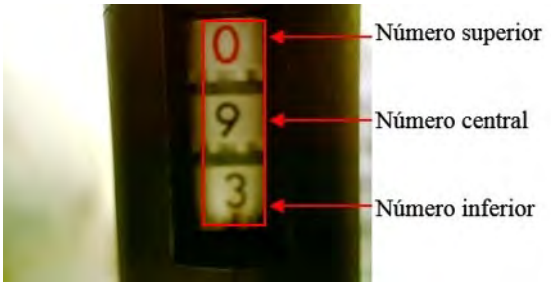

Figura 3. Indicador de volumen de la micropipeta de 100 µL de capacidad.

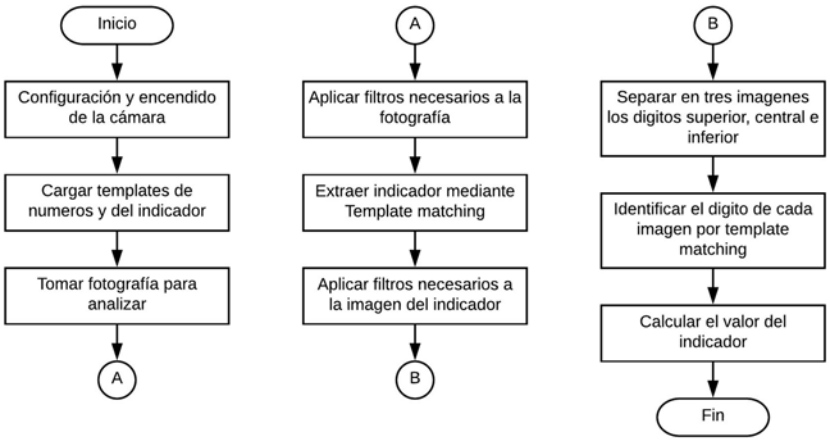

Figura 4. Diagrama de flujo para la lectura del volumen de la micropipeta.

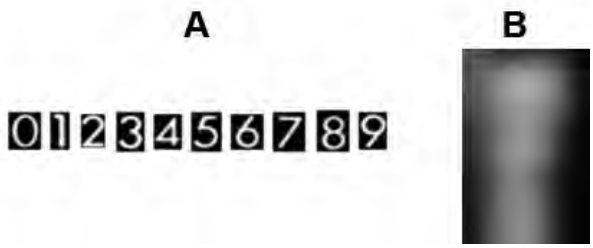

Figura 5. Plantillas implementadas. (A) En la detección de los dígitos. (B) En la extracción del indicador de volumen.

Resultados y discusión

El algoritmo se probó montando la micropipeta y la cámara en soportes universales mediante pinzas de 3 dedos. La cámara se colocó frente al indicador a una distancia de 1cm como se muestra en la Figura 6, el código de programación puede ser consultado al final del documento.

La iluminación del indicador de volumen de la micropipeta fue optimizado por un arreglo de LEDs de baja potencia en una configuración de 4x4 alimentados a un voltaje de 5 V, con lo cual se tuvo una identificación exitosa de los dígitos del indicador de la micropipeta. A continuación, en la figura 7, se muestran las imágenes obtenidas durante el procedimiento de extracción del volumen de la micropipeta.

En esta primera etapa de determinación del volumen de una micropipeta, el uso de la cámara Raspberry Pi en conjunto con la micro computadora *Raspberry Pi* 3B se logró el procesamiento de imágenes de manera satisfactoria.

Para realizar una lectura por medio del *script* de *python* "*Tmatching.py*" se debe de realizar lo siguiente: (1) Conectar la cámara a la *Raspberry Pi*. (2) Colocar la cámara de la micropipeta a 1cm de distancia del indicador de volumen. (3) Enfocar manualmente la cámara para captar una imagen nítida. (4) Asegurar la correcta iluminación hacia el indicador de volumen. (5) Encender la *Raspberry Pi* 3 y por medio de la terminal, navegar hasta la carpeta que contiene el *script* "*Tmatching.py*". (6) Al ejecutar el *script* "*Tmatching.py*", mostrará las imágenes del proceso.

Figura 6. Montaje del sistema de detección del volumen de la micropipeta.

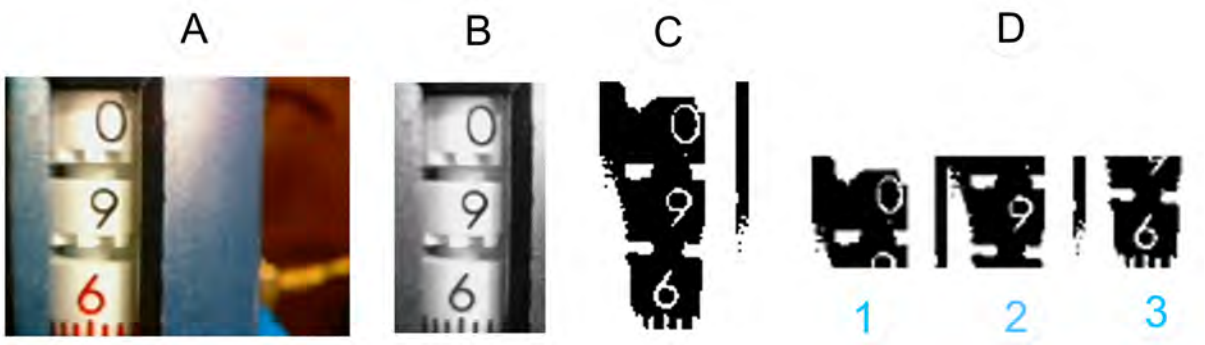

Figura 7. Proceso de identificación del volumen de la micropipeta. (A) Foto capturada para iniciar la detección. (B) Indicador del volumen recortado. (C) Conversión de la imagen a binario. (D) Dígitos extraídos para la detección, 1. centenas 2.decenas. 3. Unidades (Fuente: Elaboración propia).

Conclusión

Con base en los resultados de esta primera etapa de visión artificial, el proceso de segmentación e identificación de los dígitos de la micropipeta se realizó con éxito.

Esta técnica es muy sensible a diversas variaciones adjudicadas a la localización, intensidad y tipo de fuente luminosa, la distancia entre la cámara y el indicador del volumen de la micropipeta.

La distancia focal óptima para el correcto reconocimiento de la lectura entre el lente de la cámara y el indicador de la micropipeta fue de 1 cm.

Agradecimientos

A la Universidad del Mar campus Puerto Escondido por las facilidades otorgadas para realizar el presente trabajo y a los revisores anónimos por sus observaciones realizadas en el mejoramiento del presente manuscrito.

Referencias

Adamo, F., Attivissimo, F., Cavone, G., Giaquinto, N., & Lanzolla, A. M. L. 2006. Artificial Vision Inspection Applied To Leather Quality Control. In 13th International Conference on Pattern Recognition 2: 17-22. https://www.imeko.org/publications/wc-2006/PWC-2006-TC21-015u.pdf

Andrew Alliance 2020. Andrew liquid handling robot. Recuperado el 25 de marzo de 2020, de: https://www.andrewalliance.com/

ASSIST PLUS 2020. Integra Biosciences AG. Laboratory pipetting robot ASSIST PLUS Recuperado el 28 de marzo de 2020, de:https://www.integra-biosciences.com/japan/en/pipetting-robots/assist-plus

Bautista, R., Constante, P., Gordon, A., & Mendoza, D. 2019. Diseño e implementación de un sistema de visión artificial para análisis de datos NDVI en imágenes espectrales de cultivos de brócoli obtenidos mediante una aeronave pilotada remotamente. Infociencia 12(1): 30-35. https://doi.org/10.24133/infociencia.v12i1.1230

Bergamini, L., Sposato, M., Pellicciari, M., Peruzzini, M., Calderara, S., & Schmidt, J. 2020. Deep learning-based method for vision-guided robotic grasping of unknown objects. Advanced Engineering Informatics 44: 101052. https://doi.org/10.1016/j.aei.2020.101052

Brunelli, R. 2009. Template matching techniques in computer vision: Theory and practice. John Wiley & Sons. United Kingdom. 348 pp. https://doi.org/10.1002/9780470744055

Constante, P., Gordon, A., Chang, O., Pruna, E., Acuna, F., & Escobar, I. 2016. Artificial vision techniques to optimize Strawberry's industrial classification. IEEE Latin America Transactions 14(6): 2576-2581. https://doi.org/10.1109/tla.2016.7555221

Cubero, S., Aleixos, N., Moltó, E., Gómez, J., & Blasco, J. 2011. Advances in machine vision applications for automatic inspection and quality evaluation of fruits and vegetables. Food and bioprocess technology 4(4): 487-504. https://doi.org/10.1007/s11947-010-0411-8

Cuevas, E., Osuna V., & Oliva, D. 2017. Template Matching. In: Evolutionary Computation Techniques: A Comparative Perspective. Studies in Computational Intelligence. (686). Springer. Berlin. 222 pp. https://doi.org/10.1007/978-3-319-51109-2_4

Donat, W. 2014. The Raspberry Pi and the Arduino. In: Learn Raspberry Pi Programming with Python, Apress, Berkeley, CA. 215-226. https://doi.org/10.1007/978-1-4302-6425-5_14

EpMotion® 96 2020. Liquid Handling Workstations, Automated Pipetting. Eppendorf International. Recuperado 25 de marzo de 2020, de https://online-shop.eppendorf.com/OC-en/Automated-Pipetting-44509/Liquid-Handling-Workstations-44510/epMotion96-and-96xl-PF-91449.html

Eppendorf 2020. Eppendorf Research® Plus. Manual de instrucciones https://www.eppendorf.com/product-media/doc/es/174967/Eppendorf_Liquid-Handling_Operating-manual_Research-plus.pdf

González, A., Martínez, F., Pernía, A., Alba, F., Castejón, M., Ordieres, J., & Vergara, E. 2006. Técnicas y algoritmos básicos de visión artificial. La Rioja, Universidad de la Rioja.92 pp. https://publicaciones.unirioja.es/catalogo/monografias/mdi24.shtml

GIMP 2019. GNU Image Manipulation Program V 2.10. Consultado 19 junio 2019 de: https://www.gimp.org

Goyal A., & Meenpal T. 2020. Template Matching for Kinship Verification in the Wild. In: Computational Intelligence in Pattern Recognition. 255-265. Springer, Singapore. 558 pp. https://doi.org/10.1007/978-981-13-9042-5_22

Lee, L. M., & Liu, A. P. (2014). The application of micropipette aspiration in molecular mechanics of single cells. Journal of nanotechnology in engineering and medicine 5(4): 040902. https://doi.org/10.1115/1.4029936

León, R., Durán, Y., Quina, L., Yapo, E., & León, A. 2020. Visión artificial en reconocimiento de patrones para clasificación de frutas en agronegocios. PURIQ 2(2): 166-180. https://doi.org/10.37073/puriq.2.2.76

Márquez Díaz, J. E. 2020. Deep Artificial Vision Applied to the Early Identification of Non-Melanoma Cancer and Actinic Keratosis. Computación y Sistemas 24(2): https://doi.org/10.13053/cys-24-2-2901

Mettler Toledo 2014. Manual de pipeteo para ciencias de la vida. Disponible en: https://www.mt.com/dam/LabDiv/Campaigns/life_science_2014/downloads/pipetting_handbook_en.pdf

Opentrons 2021. Open source lab automation. Recuperado 2 de marzo de 2021, de https://opentrons.com

Ortiz, Ó., Amestoy, M., & Carrero, J. 2020. Positioning measurement using a new artificial vision algorithm in LabVIEW based on the analysis of images on an LCD screen. The International Journal of Advanced Manufacturing Technology 109(1-2): 155-170. https://doi.org/10.1007/s00170-020-05497-2

Paulitschke, M., & Nash, G. 1993. Micropipette methods for analysing blood cell rheology and their application to clinical research. Clinical Hemorheology and Microcirculation 13(4): 407-434. https://doi.org/10.3233/ch-1993-13401

Raspberry Pi 2019. Página oficial de Raspberry Pi. Recuperado 19 de junio de 2019, de: https://www.raspberrypi.org/products/

Shi, F., Wang, J., Shi, J., Wu, Z., Wang, Q., Tang, Z., ... & Shen, D. 2020. Review of artificial intelligence techniques in imaging data acquisition, segmentation and diagnosis for covid-19. IEEE reviews in biomedical engineering 1-11 https://doi.org/10.1109/rbme.2020.2987975

Upton, E., & Halfacree, G. 2016. Meet the Raspberry Pi. In: Raspberry Pi® User Guide. 11-22. John Wiley & Sons United Kingdom. 293 pp. https://doi.org/10.1002/9781119415572.ch1

Valencia, J., Cruz, J., Caicedo, L., & Chamorro, C. 2014. Extracción de características del iris como mecanismo de identificación biométrica. Revista Virtual Universidad Católica del Norte 42: 182-196. Disponible en: https://www.redalyc.org/articulo.oa?id=1942

Vergis, S., Komianos, V., Tsoumanis, G., Tsipis, A., & Oikonomou, K. 2020. A Low-Cost Vehicular Traffic Monitoring System Using Fog Computing. Smart Cities 3(1): 138-156. https://doi.org/10.3390/smartcities3010008

Material complementario

```
#Librerias necesarias
import sys
import cv2
import picamera
import numpy as sleep
#Function para detectar un dígito por separado
#img: imagen en la cual se detectará el número
#Tptl: Arreglo con todos los templates de números necesarias
def Detectar(img, Tptl):
        i=0
        detectado = false
#Realiza comparación con todos los números
#Hasta encontrar un Match
while (not detectado and i<10):
        #Detección de un solo número
        result = cv2.matchTemplate (img, Tplt[i] , cv2 . TM_CCOEFF_NORMED)
        loc = np.where (result>=0.7)
        detectado = np.size(loc)>0
        #Si se detecta un match el ciclo while se detiene
        if detectado:
                break
        else:
                i += 1
#Si hay una detección devuelve el número detectado
#caso contario devuelve un 0
        if detectado:
                return i
        else:
                return 0
#Function Main
def main ( ):
#Guarda el nombre de la imagen en la cual se detecta un número
Image_name = "sample.jpg"
#Arreglo de todos los Templates numéricos, cada imagen se ubica
#en la posición del arreglo correspondiente a su valor numérico
Tptl = []
Tptl.append(cv2.imread( "T0_bin.jpg" ,cv2.IMREAD_GRAYSCALE))
Tptl.append(cv2.imread( "T1_bin.jpg" ,cv2.IMREAD_GRAYSCALE))
Tptl.append(cv2.imread( "T2_bin.jpg" ,cv2.IMREAD_GRAYSCALE))
Tptl.append(cv2.imread( "T3_bin.jpg" ,cv2.IMREAD_GRAYSCALE))
```

```
Tptl.append(cv2.imread( “T4_bin.jpg” ,cv2.IMREAD_GRAYSCALE))
Tptl.append(cv2.imread( “T5_bin.jpg” ,cv2.IMREAD_GRAYSCALE))
Tptl.append(cv2.imread( “T6_bin.jpg” ,cv2.IMREAD_GRAYSCALE))
Tptl.append(cv2.imread( “T7_bin.jpg” ,cv2.IMREAD_GRAYSCALE))
Tptl.append(cv2.imread( “T8_bin.jpg” ,cv2.IMREAD_GRAYSCALE))
Tptl.append(cv2.imread( “T9_bin.jpg” ,cv2.IMREAD_GRAYSCALE))
#Se lee el template para detectar el indicador de volumen en escala de grises
Tblur = cv2. Imread (“T_blur.jpg” , cv2. IMREAD_GRAYSCALE)
        x, y = Tblur.shape[::-1]
#Se captura la imagen por medio de la cámara y se guarda en un archivo
#posteriormente se lee a una variable compatible con el proceso de Matching
camera.capture(image_name)
        img = cv2.imread(image_name)
        img_gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
#Se realza el Match entre el template Tblur y la imagen para extraer el indicador
#de la fotografía tomada
crop_res = cv2.matchTemplate(img_gray, Tblur, cv2.TM_CCOEFF_NORMED)
crop_loc = np.where(crop_res>=0.6)
#Se extrae el indicador de volumen de la imagen
        for pt in zip(*crop_loc[::-1 ]):
                crop_img = img_gray[pt[1]:pt[1] + y,pt[0]:pt[0] + x]
#Se convierte a binario la imagen para una mejor detección de los números
Thresh = cv2.threshold(crop_img, 0, 255, cv2. TRESH_BINARY_INV | cv2. TRESH_OTSU) [1]
#La imagen se divide en tres para poder aislar los números superior, central e #inferior
centenas = thresh [0: y/2, 0::]
decenas =  thresh [y/4:3*y/4, 0::]
unidades = thresh [y/2::,  0::]
#Se detecta y convierte a valor numérico cada dígito
value = 0
value += Detectar (centenas, Tptl)*100
value += Detectar (decenas, Tptl)*10
value += Detectar (unidades, Tptl)*1
#Configuración y encendido de la cámara
with picamera. Picamera ( ) as camera:
#Resolución máxima
camera.resolution = (128, 96)
#Cambio de los parámetros para fotos consistentes
camera.iso = 600
sleep (2)
camera.shutter_speed = camera.exposure_speed
camera.exposure_mode = ´off´
        g = camera.awb_gains
```

```
        camera.awb_mode = ´off ´
        camera.awb_gains = g
        print "ready"
        if __name__ == "_main_":
                main( )
```

Script ImageTest.py

```
import picamera
from time import sleep
def main ( ):
        camera.capture(´sample.jpg´)
#Configuración y encendido de la cámara
with picamera. Picamera ( ) as camera:
#Fotos en blanco y negro
#camera.color_effects = (128,128)
#Resolución máxima
camera.resolution = (128, 96)
#Cambio de los parámetros para fotos consistentes
camera.iso  =  600
sleep (2)
camera.shutter_speed = camera.exposure_speed
camera.exposure_mode = ´off´
        g = camera.awb_gains
        camera.awb_mode = ´off ´
        camera.awb_gains = g
        print "ready"
        if __name__ == "_main_":
                main( )
```

Script Bin.py

```
import sys
import cv2
import numpy as np

def main ( ):
        imahe_name = sys.argv[1] + ".jpg"
        img = cv2.imread(image_name,cv2.IMREAD_GRAYSCALE)
thresh = cv2.threshold(crop_img, 0, 255, cv2. TRESH_BINARY_INV | cv2. TRESH_OTSU) [1]
        bin_name = sys.argv[1] + "_bin.jpg"
        cv2.imwrite(bin_name, thresh)
if __name__ ==  "__main__":
        main ( )
```