ISSN 1870-4069

Detector de precios de gasolina basado en visión artificial y computación en la nube

Mircel K. García Rodríguez[1], Osslan O. VergaraVillegas[1], Ivón O. Benítez González[2]

[1] Universidad Autónoma de Ciudad Juárez, México

[2] Universidad Tecnológica de La Habana José Antonio Echeverría, Cuba

al183283@alumnos.uacj.mx, overgara@uacj.mx, ivonoristelabg@gmail.com

Resumen. El reconocimiento óptico de caracteres dentro de los sistemas de visión permite automatizar información y procesos en los que con más frecuencia la intervención del ser humano es cuestionada. Tal es el caso de la actualización manual de los precios de la gasolina en las estaciones de servicio de combustibles que ha provocado incongruencias entre el precio real y el publicado en las plataformas de la Comisión Reguladora de Energía (CRE) en México. El objetivo del artículo es presentar un detector automático de precios para las gasolinas que se ofertan en una estación de servicio usando una cámara digital, así como un procedimiento para enviar la información a la nube de Microsoft Azure. Así, será posible mostrar a los usuarios de forma automática el precio exacto de la gasolina a través de una aplicación móvil o una página web. La metodología se divide en 2 etapas: i) implementar el detector de precios de la gasolina, y ii) desplegar una función en la nube de Azure y almacenar el precio de la gasolina en una base de datos. De acuerdo con los resultados obtenidos se concluye que el detector reconoce los precios de la gasolina con una exactitud del 90%, lo cual es competitivo con los resultados presentados en la literatura. Además, las transferencias a la nube Azure de los precios detectados fueron 100% exitosas si la conexión a Internet está activa.

Palabras clave: Computación en la Nube, Reconocimiento Óptico de Caracteres, Flexibilidad, Escalabilidad.

Gasoline Price Detector Based on Machine Vision and Cloud Computing

Abstract. Optical character recognition within vision systems allows to automate information and processes in which the intervention of the human being is frequently questioned. The case of the manual update of gasoline prices at fuel service stations has caused inconsistencies between the real price and the one published on platforms by the Energy Regulatory Commission (ERC) in Mexico.

This paper aims to present an automatic price detector for gasoline offered at a service station using a digital camera and a procedure to send the information to the Microsoft Azure cloud. Thus, it will be possible to show automatically the users the exact price of gasoline through a mobile application or a web page. The methodology comprises two stages: i) implement the gasoline price detector, and ii) deploy a function in the Azure cloud and store the price of the gasoline in a database. According to the results obtained, we concluded that the proposed detector recognizes gasoline prices with an accuracy of 90%, which is competitive with the results presented in the literature. Furthermore, transfers to the Azure cloud of the detected prices were 100% successful if the Internet connection is active.

Keywords: Cloud Computing, Optical Character Recognition, Flexibility, Scalability.

1. Introducción

Muy jocoso y antagónicamente cautivador, resulta pensar que "la fábrica del futuro tendrá dos empleados: un humano y un perro. La labor del humano será dar de comer al perro y la del perro, evitar que el humano toque los sistemas automatizados" [1]. En efecto, una de las tendencias actuales de la tecnología se basa en los pilares de la industria 4.0 [2-4]. La industria 4.0 o Cuarta Revolución Industrial, es un conjunto de conceptos con influencia sustancial en el sector manufacturero [5]. Uno de los conceptos más importantes y que las diversas industrias buscan adoptar es la computación basada en "la nube" [3, 4].

La revolución industrial 4.0 ha capturado la atención de industrias en todo el mundo, incluyendo al sector petrolero y de gas [6]. Dicho sector se ha incorporado a la nueva era de la informatización y las comunicaciones. Sin embargo, la transición hacia la industria 4.0 ha sido paulatina y conservadora respecto a los servicios financieros y el sector de las telecomunicaciones [6]. Al parecer, todo apunta a que la transición paulatina se debe fundamentalmente a problemas de seguridad y protección de los datos que aún se presentan como brechas tecnológicas.

Los sistemas de visión han ocupado un puesto importante dentro de los nuevos paradigmas actuales que se han establecido. Los problemas en áreas como la biotecnología, la robótica y la realidad aumentada han protagonizado atractivas soluciones basadas en visión cada vez más complejas y sofisticadas [7, 8]. Una de las industrias que ha implementado soluciones basadas en sistemas de visión es la transportación [9]. En algunos países como Estados Unidos se ha planteado un desafío para contrarrestar el aumento en los accidentes y el incremento en las pérdidas económicas, especialmente las devenidas del aumento en el consumo de combustibles [9]. Una alternativa para combatir dicho desafío es implementar sistemas con sensores en los lugares donde se puede observar mayor actividad del transporte terrestre. Sin embargo ¿qué utilidad, por ejemplo, pudieran tener dichos sensores en las gasolineras que forman parte de la cadena de valor de la industria petrolera más allá de su rol tradicional como complemento para la vigilancia y la supervisión del proceso?

Aun inmersos en una nueva era industrial, en los establecimientos minoristas de distribución de combustibles no se han implementado simultánea y masivamente, las tendencias actuales de la tecnología, especialmente los sistemas de visión y la

computación en la nube. Particularmente, la información relativa al precio del producto en una gasolinera actualmente no está disponible en la red pública de forma automatizada, y el usuario sólo la puede verificar in situ. Queda de parte de propietarios e inversionistas comprender que la descentralización de los precios de la gasolina en muchos países está inclinando cada vez más la balanza del lado del cliente [10], en donde es posible apostar por una cartera de servicios adyacentes o bien reinvertir en nuevos activos que motiven y estimulen el interés del consumidor.

Los datos que se pueden obtener con un detector ofrecen información relevante a los consumidores como precios y disponibilidad de productos. Es posible incrementar el poder de decisión de los clientes poniendo a su disposición dicha información a través de una aplicación móvil o un sitio web. Con el uso de los recursos que ofrece la nube pública es posible intercambiar información con el usuario rápidamente.

El resto del artículo se encuentra organizado de la siguiente manera: en la Sección 2 se presentan los métodos utilizados para la generación del detector de precios de gasolina. En la Sección 3, se presenta la validación del sistema propuesto utilizando 25 imágenes. Además, se presenta la metodología para almacenar los resultados obtenidos en una base de datos en la nube. Por último, en la Sección 4, se presentan las conclusiones del trabajo de investigación.

2. Materiales y métodos

La metodología para implementar el detector de precios de la gasolina se muestra en la Fig. 1. Además, en la Tabla 1 se presentan los parámetros y los rangos que se utilizaron para validar el sistema. El primer paso consiste en obtener una representación RGB (rojo, verde y azul) de una imagen que representa un cartel de precios real. El preprocesamiento incluye una conversión a escala de grises que reduce a 256 colores la representación matricial de la instantánea. El filtrado se aplica para eliminar ruido convolucionando una matriz cuadrada de Gauss de dimensión R=5 (parámetros $\boldsymbol{blur_1}$ y $\boldsymbol{blur_2}$ en la Tabla 1) con cada pixel de la imagen en gris [11]. La imagen resultante del filtrado se pasa al detector de bordes *Canny*, que utiliza la dirección del gradiente de la intensidad de los pixeles, la cual es perpendicular a los bordes [12]. Además, se emplea umbralización por histéresis para determinar qué aristas son realmente bordes. Finalmente, una función basada en la mediana de la intensidad de los pixeles calcula los valores máximo y mínimo del umbral.

La etapa del seccionado de los precios se implementa usando las coordenadas X e Y del punto inicial y final (polígono de cuatro lados que representa la sección de cada precio). El seccionado de la imagen se implementa dentro de un ciclo iterativo *for* y la cantidad de secciones o ciclos depende de la posición de los precios en la imagen. Una vez que se han seccionado los precios, se procede al proceso de binarización adaptativa, por lo que se determina el umbral de un píxel en función de su vecindad. Así es posible obtener umbrales para diferentes regiones de la misma imagen, lo que brinda mejores resultados cuando la iluminación es variable [13]. El resultado permite obtener una imagen cuyo fondo es negro con los caracteres o precios de color blanco (matriz de 1 y 0 con binarización invertida).

Después, se dibuja un marco para eliminar falsos contornos externos. El grosor del marco está definido por el parámetro *lineThickness* de la Tabla 1.

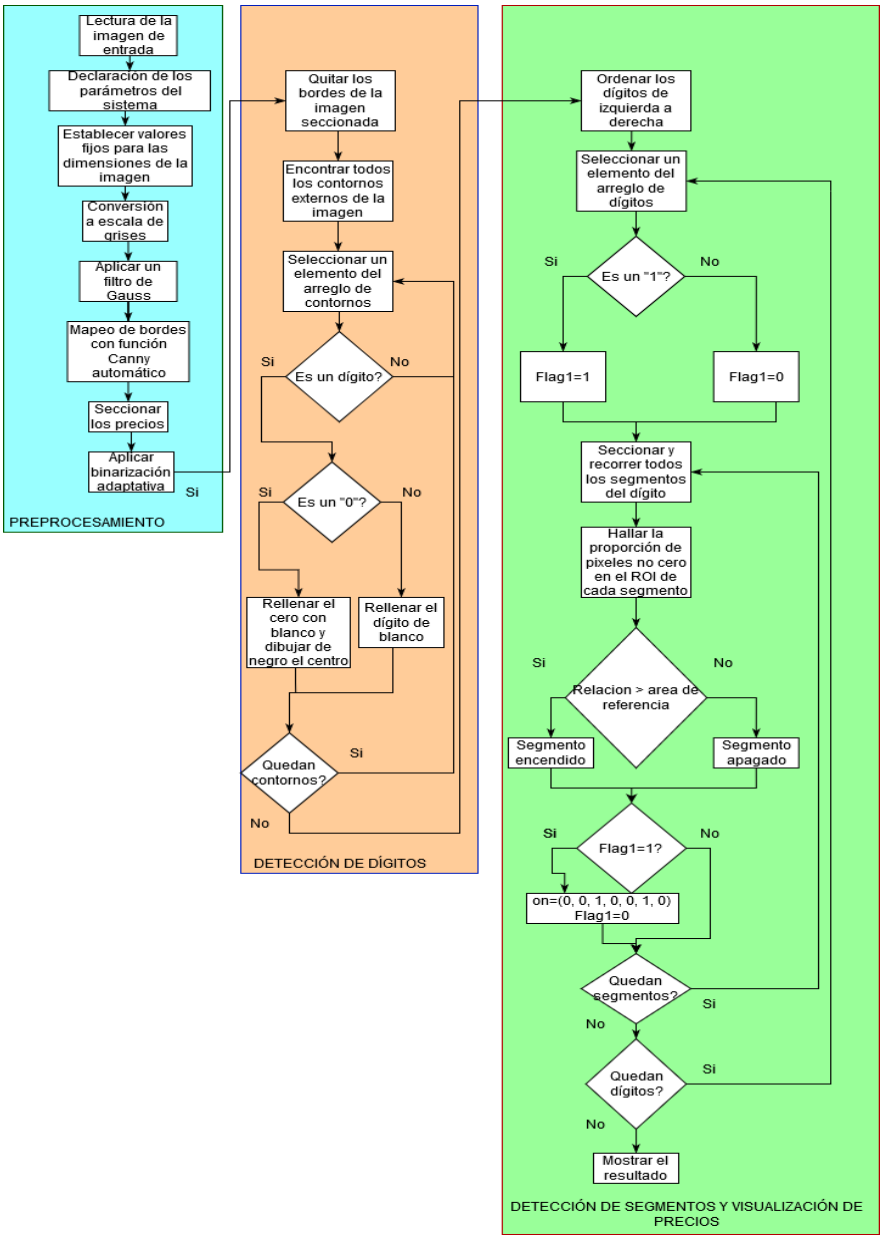


Fig. 1. Diagrama de flujo para el detector de precios de la gasolina.

La etapa de detección y clasificación comienza después de encontrar todos los contornos en la imagen resultante. Posteriormente, se implementó una rutina para encontrar todos los contornos externos y no hacer ninguna aproximación de los puntos que los forman.

Tabla 1. Resumen de parámetros y valores utilizados para implementar el detector de precios.

Parámetros	Condición ajustable	Mínimo valor	Máximo valor
$\boldsymbol{arearef_{min}}$	si	0.31	-
$\boldsymbol{arearef_0}$	si	0.07	-
$\boldsymbol{dw_{ref}}$	si	0.25	-
$\boldsymbol{dh_{ref}}$	si	0.15	-
$\boldsymbol{w_1}$	si	0	25
$\boldsymbol{flag_1}$	no	0	1
$\boldsymbol{Digits_{Lookup}}$	no	1 1 1 0 1 1 1 : 0 0 0 1 0 0 1 0 : 1 1 0 1 1 1 0 1 : 2 1 0 1 1 0 1 1 : 3 0 1 1 1 0 1 0 : 4 1 1 0 1 0 1 1 : 5 1 1 0 1 1 1 1 : 6 1 0 1 0 0 1 0 : 7 1 1 1 1 1 1 1 : 8 1 1 1 1 0 1 1 : 9	-
$\boldsymbol{thresh_1}$	si	3	5
$\boldsymbol{blur_1}$	si	25	45
$\boldsymbol{blur_2}$	si	5	25
$\boldsymbol{color_{fill}}$	no	255,255,255	-
$\boldsymbol{color_{fill\_cero}}$	no	0,0,0	-
$\boldsymbol{w_{descriptor}}$	si	8	70
$\boldsymbol{h_{descriptor}}$	si	45	105
$\boldsymbol{line_{Thickness}}$	si	10	20

El cuadro delimitador (*bounding box*) de cada contorno se obtiene por medio de una rutina que aproxima un rectángulo alrededor de la imagen binaria, y por medio de un ciclo *for* se recorren todos los bordes encontrados.

Cada contorno es clasificado en dígitos o no, utilizando un descriptor basado en las dimensiones del delimitador. El descriptor evalúa si las dimensiones de (ancho (w) y alto (h)) están dentro de los valores establecidos en la Tabla 1 ($w_{descriptor}$ y $h_{descriptor}$). Si la condición se cumple se considera que el contorno actual es un dígito. Si el dígito es un cero, se cumplirá que la relación $\frac{pixeles\ centrales\ no\ cero}{area\ del\ segmento\ central} > arearef_0 = 0.07$. Se utiliza una rutina para rellenar el espacio del cero con blanco y posteriormente dibujar de negro solo el centro, como se muestra en la Fig. 2. Si el dígito no es un cero, entonces se rellena el interior de todos los contornos de blanco.

Todos los dígitos encontrados son ordenados de izquierda a derecha y son nuevamente seccionados para obtener todos los segmentos correspondientes (seccionado de segmentos del dígito). Para ello se utilizan los parámetros dw_{ref} y dh_{ref} que representan porcentajes del ancho y alto del cuadro delimitador de cada dígito.

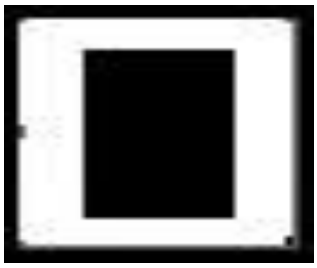


Fig. 2. Resultado del proceso de rellenado del dígito cero.

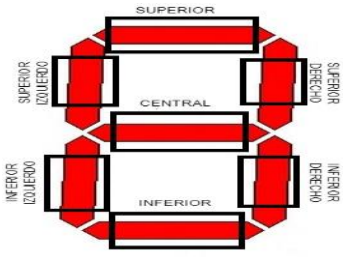


Fig. 3. Seccionado de los segmentos de cada dígito.

El nuevo seccionado permite obtener regiones o subsecciones por separado para cada uno de los siete segmentos del dígito, es decir: superior, superior izquierdo, superior derecho, central, inferior izquierdo, inferior derecho e inferior tal y como se muestra en la Fig. 3. Con cada segmento se implementa un ciclo iterativo para determinar cuáles pixeles tienen valor 0 y cuáles 1. Debe destacarse que como la umbralización previa fue de tipo invertida, un segmento encendido tiene el color blanco, y sus bits son todos uno. A partir de una relación entre la cantidad de pixeles no equivalentes a cero y el área de la subsección, es posible conocer si el segmento está o no encendido. La cantidad de pixeles no equivalentes a cero se obtiene con la operación aritmética de suma de bits que no son cero.

La relación se puede ajustar con el parámetro $\boldsymbol{arearef_{min}}$ (Tabla 1). Cuando la relación es superior al valor del parámetro, significa que más del 31% de los pixeles en la sección tienen valor uno, por lo que se asume que el segmento está encendido y se fuerza un bit a 1 (on[i]= 1). Particularmente, cuando se trata del dígito 1 se evalúa la condición ($w_1 < 25$ según la Tabla 1) y se establece el parámetro $flag_1 = 1$. Así, en dicho caso es posible además establecer que: $on = 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0$ (empaquetado binario del 1 escrito a 7 segmentos).

Luego, se recorre el arreglo de segmentos y se repite el proceso para los 6 segmentos restantes. Cuando se completan todas las iteraciones se busca el resultado en un arreglo de dígitos ($Digits_{Lookup}$ dígitos de siete segmentos empaquetados en formato binario). Una vez que los dígitos son identificados dentro del arreglo se muestra el resultado y se almacena en una variable. En la Fig. 4 se muestra una secuencia de imágenes obtenidas al aplicar el método de detección a un cartel de precios de Pemex. Las imágenes superiores corresponden a la etapa de preprocesamiento y las inferiores muestran el resultado de la detección y visualización de los precios de la gasolina.

2.1. Recolección de imágenes para la validación del detector de precios

Las 25 imágenes utilizadas para validar el detector de precios se obtuvieron de gasolineras en Ciudad Juárez, bajo condiciones de ruido críticas. Las imágenes contienen los precios de distintos proveedores. El número de establecimientos de venta en el municipio es 18. Además, con el grupo de imágenes obtenido, es posible someter al detector de precios a condiciones extremas. Algunas de las instantáneas utilizadas en la validación se muestran en la Fig. 5.

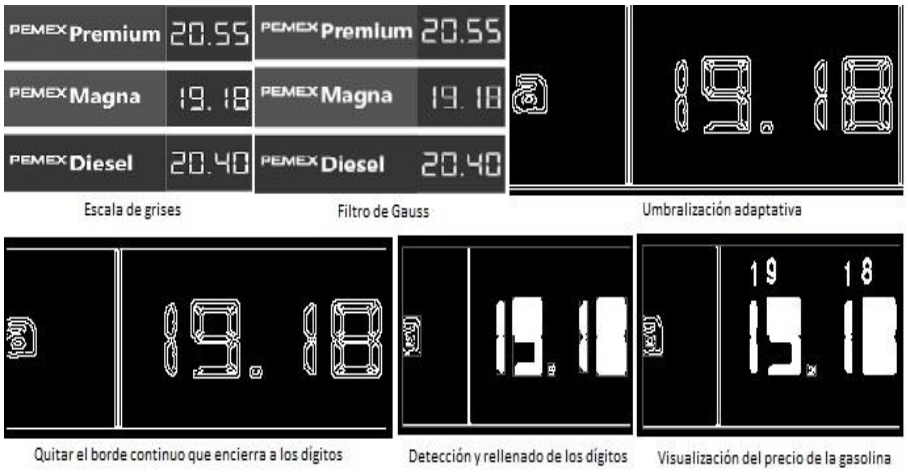


Fig. 4. Resultados del detector de precios.

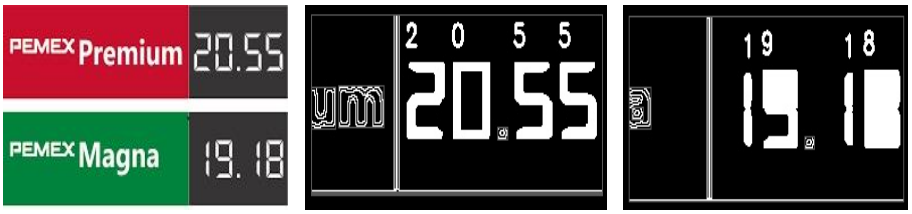
Fig. 5. Ejemplo de imágenes utilizadas en la validación del detector.

2.2. Detección de los precios de la gasolina para un caso base

Después de implementar el detector se procede a la validación. Para ello se utiliza una construcción digital de un cartel de precios real la cual se utiliza como caso base (ver Fig. 6). Una vez que el detector converge al resultado esperado se valida con 25 imágenes reales y los resultados se muestran en la Sección 3.

El resultado obtenido para los precios de la gasolina Premium y Magna respectivamente se muestra en la Fig. 6, sección b y c. Note que se representan exactamente las cantidades o precios impresos en el cartel de la Fig. 6, sección a. Justo encima de cada dígito se ha representado el valor correspondiente obtenido del arreglo $\boldsymbol{Digits_{Lookup}}$. Los dígitos han sido rellenados con pixeles equivalentes a uno.

Es importante destacar que en ambas figuras ha sido necesario eliminar el marco o borde que encerraba a los precios. Dicho marco hubiese impedido la detección de contornos con la configuración CV_RETR_EXTERNAL. Los precios obtenidos con el detector se asignan a variables para ser posteriormente gestionados y almacenados en una infraestructura de nube completa como Microsoft Azure.



a) b) c)

Fig. 6. Ejemplo de detección de precios. a) Cartel usado como base, b) detección del precio para gasolina Premium, c) detección del precio para gasolina Magna.

Tabla 2. Tabla *dbo.gasolinera* dentro de la DB en Azure.

id	PrecioMagna	PrecioPremium

2.3. Metodología para almacenar datos en la nube Azure

El envío de datos a la nube permite mostrar los precios a clientes potenciales sin la necesidad de crear una infraestructura física. El método de presentación de la información puede ser escalable y automatizado a partir de: i) Crear una cuenta de Azure mediante el vínculo: azure.microsoft.com, ii) Crear una Base de Datos (BD) en Azure usando el editor de consultas SQL. Dentro de la BD se crea una tabla con el formato que se muestra en la Tabla 2, iii) Programar y desplegar una función en lenguaje Java en la nube desde el entorno de desarrollo integrado (IDE) Visual Studio Code (VS Code) para almacenar los precios en la BD, y iv) Programar un método en el cliente remoto para invocar la función desplegada en la nube Azure.

La función en lenguaje Java para almacenar los precios en la nube se programa en VS Code y se despliega usando la línea de comandos en la terminal del IDE. Cabe destacar que la función se despliega en modo solo lectura por lo que los cambios se deben hacer necesariamente desde el VS Code (flexibilidad reducida). Para desplegar la función es necesario instalar Maven en la máquina local. Cuando se ha completado el código de la función es posible desplegarla en Azure escribiendo los comandos: az login (paso de autentificación) y mvn azure-functions:deploy (despliegue).

Finalmente, se debe programar un método en el cliente remoto (detector de precios) para invocar a la función Java desplegada en la nube Azure. El método utiliza el formato *Java Script Object Notation* (JSON) para enviar los precios de la gasolina a la nube y desencadenar el código de la función. Por ello es importante disponer de la dirección url de la función, la cual puede ser obtenida en el portal de Azure.

3. Experimentación y resultados

Los resultados obtenidos en la validación del detector se muestran en la Tabla 3. Durante la fase de experimentación se realizaron 50 corridas base para obtener el precio de la gasolina Magna y Premium respectivamente. La efectividad del detector corresponde con los aciertos totales en todas las imágenes. En el proceso de detección se empleó la función de rellenado de espacios debido a que se detectó que algunos

PrecioMagna	PrecioPremium
12.35	14.57
19.59	20

Fig. 7. Consulta a la DB en Azure.

Tabla 3. Resultados obtenidos en la validación del detector.

No. de imágenes	Aciertos Magna	Aciertos Premium	Efectividad
25	23	22	90%

Tabla 4. Transferencia de datos a la nube.

Envío de paquetes de datos a la nube	Cantidades totales enviadas a Azure	Cantidades recibidas en Azure
Total de transferencia para el detector de precios (paquete Magna+Premium)	23 paquetes (45 precios)	45 precios

dígitos binarizados presentaban "agujeros" que impedían estandarizar el descriptor de segmentos. El caso del número 1 se implementó con la bandera $flag_1$ dado que al obtener el delimitador del contorno, sólo se lograban capturar dos segmentos del dígito. Por lo que, era imposible clasificarlo correctamente. Sin embargo, no fue necesario aumentar el vector de características o completar el delimitador con una reconstrucción para abarcar los 7 segmentos. En la parametrización del detector, las dimensiones de campana y el tamaño de la vecindad de pixeles $thresh_1$ se obtuvieron de forma empírica a partir de la evaluación del conjunto de imágenes. En el caso del detector *Canny*, los valores de los umbrales mínimo y máximo se obtuvieron con una umbralización automática que encuentra el mejor valor del umbral utilizando la mediana de la intensidad de los pixeles.

Finalmente, para validar el despliegue de los precios de la gasolina en la DB en Azure, se enviaron dos pares de precios arbitrarios tal y como se muestra en la Fig. 7. La transferencia de los precios fue exitosa según los resultados obtenidos en el editor de consultas de Azure. Luego y según los datos mostrados en la Tabla 4, se enviaron a la nube los 45 precios detectados de los cuales se almacenaron en la DB el 100%.

3.1. Discusión

Las imágenes utilizadas en la validación contienen ruido y características que complejizan la detección del precio de la gasolina. En los casos que no se pudo detectar el precio de la gasolina interviene la presencia de ruido que no se pudo atenuar con una campana de Gauss de dimensiones superiores a 25 pixeles.

Por otro lado, el color rojo de algunos dígitos hace que desaparezcan cuando se binariza la imagen por lo que no se pudieron detectar los precios. Finalmente, se suscitaron algunos casos donde la presencia del punto decimal cerca del dígito afectó la segmentación y por lo tanto, no se pudo detectar el precio.

Durante la etapa de almacenamiento en la nube se comprobó que las transferencias son exitosas y fiables siempre que exista una conexión activa a Internet. Sin embargo, uno de los inconvenientes del despliegue en la nube es la necesidad de desarrollar localmente la función Java antes de publicarla. Los entornos de ejecución en el portal tienen algunas limitaciones cuando el lenguaje es diferente de C#.

4. Conclusiones

En el presente artículo se mostró una nueva tecnología en el marco de las transformaciones tecnológicas para automatizar la detección y almacenamiento en la nube del precio de la gasolina. El detector permite obtener de forma automatizada el precio con una efectividad del 90% siempre y cuando los dígitos estén escritos a 7 segmentos. La legibilidad del cartel de precios de la gasolina es un factor importante por considerar lo que supone la utilización de una cámara con buenas prestaciones. Las transferencias de datos a la nube son fiables y le proporcionan acceso ubicuo a la información relevante para los clientes.

A futuro se propone utilizar y comparar con otros métodos de detección de caracteres para evaluar el desempeño del sistema. Para mejorar el desempeño del detector se propone como trabajo futuro, probar con otros algoritmos de suavizado como: filtrado bilateral, difusión anisotrópica o filtro kuwahara. Para el caso de las fallas ante dígitos de color rojo se pretende comparar en futuras investigaciones el modelo de colores RGB con otro diferente como por ejemplo el HSV o CIELAB.

Referencias

1. Sáenz, C.C.: Industria 4.0 (Tesis de maestría), Universidad de la Rioja, España (2016)
2. Dodgson, M., Gann, D.M., Salter, A.: The management of technological innovation: strategy and practice. Oxford University Press on Demand (2008)
3. Ibarra, D.: Business model innovation through Industry 4.0: A review. Procedia Manufacturing, 22, pp. 4−10 (2018)
4. Rüßmann, M., Lorenz, M., Gerbert, P., Waldner, M., Justus, J., Engel, P., Harnisch, M.: Industry 4.0: The future of productivity and growth in manufacturing industries. Boston Consulting Group, 9, pp. 54−89 (2015)
5. Stock, T., Seliger, G.: Opportunities of sustainable manufacturing in industry 4.0. Procedia Cirp, 40, pp. 536−541 (2016)
6. Perrons, R.K., Hems, A.: Cloud computing in the upstream oil & gas industry: A proposed way forward. Energy Policy, 56, pp. 732−737 (2013)
7. Ali, L., Khan, W., Chaiyasarn, K.: Damage detection and localization in masonry structure using faster region convolutional networks. Int. J. Geomate, 17, pp. 98−105 (2019)
8. Shahid, L., Janabi-Sharifi, F., Keenan, P.: A hybrid vision-based surface coverage measurement method for robotic inspection Robot. Comput.-Integr. Manuf., 57, pp. 138−145 (2019)
9. Walid., B., Refai, H.H.: Intelligent vehicle counting and classification sensor for real-time traffic surveillance. Transactions on Intelligent Transportation Systems, 19 (2018)
10. Fernández, G., Almajano, J., García, E., Bludszuweit, H., Machín, S., Sanz, J.F.: Control structure for optimal demand-side management with a multi-technology battery storage system. In: 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), IEEE, pp. 754−759 (2019)

11. Hummel, R.A., Kimia, B., Zucker, S.W.: Deblurring Gaussian blur. Computer Vision, Graphics, and Image Processing, 38(1), pp. 66−80 (1987)
12. Bao, P., Zhang, L., Wu, X.: Canny edge detection enhancement by scale multiplication. IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(9), pp. 1485−1490 (2005)
13. Bradley, D., Roth, G.: Adaptive thresholding using the integral image. Journal of Graphics Tools, 12(2), pp. 13−21 (2007)