CAPÍTULO 1

Identificación

1.1. Introducción

Introducción

1.2. Técnicas de identificación

Técnicas de identificación

1.3. QR

Ventajas del Lenguaje C para procesamiento de imágenes

1.3.1. Identificadores QR: una realidad cotidiana.

El uso de los identificadores QR (Quick Response), es cada vez más generalizado. Últimamente, debido al incremento significativo del uso de *smart devices*, el hecho de poder contar con una cámara, cierto poder de procesamiento y por lo general hasta una conexión móvil a internet, hace que sea cada vez más frecuente encontrar aplicaciones con el poder de reconocer QRs. Comenzaron utilizándose en la industria automovolística japonesa como una solución para el trazado en la línea de producción, pero su campo de aplicación se ha diversificado y hoy en día se pueden encontrar también como identificatorios de entradas deportivas, tickets de avión, localización geográfica, vínculos a páginas web y en algunos casos también como tarjetas personales.

1.3.2. ¿Qué son los QR?

Los QRs son una extensión de los códigos de barras. Incorporan una segunda dimensión lo cual es una gran ventaja ya que pueden almacenar mucho más información. Existen distintos tipos de QR, con distintas capacidades de almacenamiento que dependen de la versión, el tipo de datos almacenados y del tipo de corrección de errores. En su versión 40 con detección de errores de nivel L, se pueden almacenar alrededor de 4300 caracteres alfanuméricos o 7000 dígitos (frente a los 20-30 dígitos del código de barras) lo cual lo hace muy flexible para cualquier tipo de aplicación de identificación.

En la Figura 1.1 se pueden ver las distintas partes que componen un QR, como por ejemplo el bloque de control, compuesto por las tres esquinas idénticas que dan información de la posición, la información de alineamiento y el patrón de sincronismo; así como también la indicación de versión, formato y la corrección de errores. Fuera de toda esa información, que podría verse como el encabezado, haciendo analogía con los paquetes de las redes de datos, se encuentran los datos propiamente dicho, que podrían verse como el cuerpo del paquete.

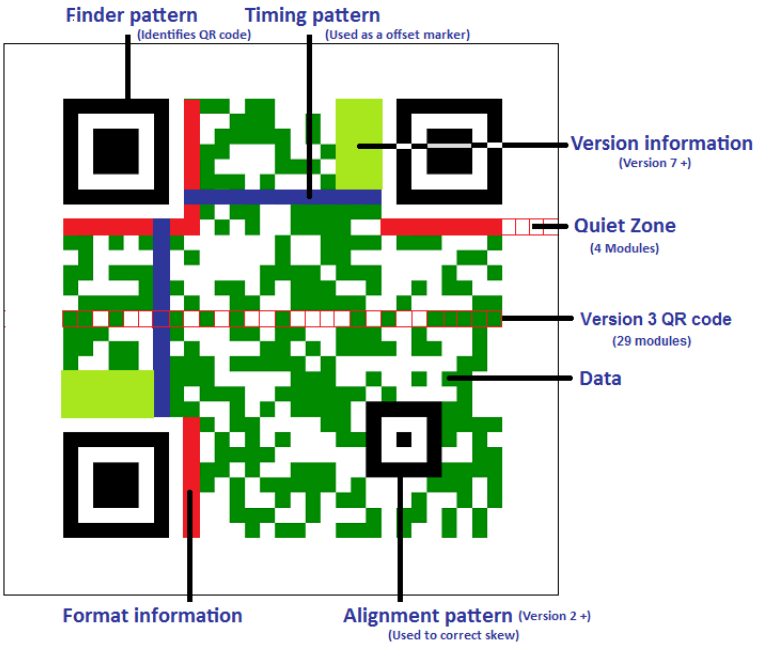

Figura 1.1: Las distintas componentes de un QR. Fuente (poner fuente).

1.3.3. Codificación y decodificación de códigos QR.

Es fácil darse cuenta que la codificación resulta mucho más sencilla que la decodificación. Para la codificación es necesario comprender el protocolo, las distintas variantes y el tipo de información que se pretende almacenar. Sin embargo, para la decodificación, además de tener que cumplir con lo anterior, es necesario contar con buenos sensores y ciertas condiciones de luminosidad y distancia que favorezcan a la cámara y se traduzcan en buenos resultados luego de la detección de errores. Si bien la plataforma es importante para lograr buenos resultados, dada una plataforma, existen variadas aplicaciones tanto para iOS como para Android que cuentan con performances bastante diferentes en función del algoritmo de procesamiento utilizado.

Debido a que el centro del presente proyecto no fue la codificación y decodificación de QRs, y que además ya existen distintas librerías que resuelven muy bien este problema, se optó por investigar varias de ellas e incorporar la más adecuada a la aplicación.

Entre todas las librerías que resuelven la decodificación, las llamadas ZXing y ZBar son quizá las más destacadas, por su popularidad, simplicidad y buena documentación para la fácil implementación. ZXing, denominada así por “Zebra Crossin”, es una librería gratis y en código abierto desarrollada en java y que tiene implementaciones que están adaptadas para otros lenguajes como

C++, Objetive-C y JRuby, entre otros.

Por su parte ZBar también tiene soporte sobre varios lenguajes y cuenta con un kit de desarrollo interesante para lograr fácilmente aplicaciones que integren el lector de QR. Se trabajó sobre el código de ejemplo que contiene la implementación de las clases principales para obtener un lector y finalmente se optó por utilizar esta librería para los fines de la aplicación. El lector del código de ejemplo consta de una clase *ReaderSampleViewController* que hereda de *UIViewController* y que implementa un protocolo llamado *ZBarReaderDelegate*. Al presionarse el botón de detección se crea una instancia de la clase *ReaderSampleViewController* y se presenta la vista previa de la cámara. Luego el protocolo se encarga de la captura y procesamiento del QR almacenando como resultado la información embebida en este en la variable denominada *ZBarReaderControllerResults*. Esta variable luego se mapea en una *hash table* con el contenido en formato *NSDictionary*. De esta manera se accede fácilmente al contenido en formato legible y es fácil de hacer una lógica de comparación y búsqueda en una base de datos.

1.3.4. Expresiones artísticas con QRs.

La opción de usar los QR de una manera distinta ha comenzado a ser notoria en los últimos tiempos. Hay quienes desafían a la información *cruda de 1s y 0s* incorporando imágenes y modificando colores y contornos en los QR tradicionales para lograr un valor estético además del funcional. Véase en la figura 1.2 un ejemplo de cómo puede lograrse el mismo resultado pero con el valor agregado de originalidad.

Figura 1.2: Ejemplo de un QR creativo. Fuente (poner fuente).

1.4. SIFT

El algoritmo SIFT, acrónimo de “Scale Invariant Feature Transform”, es un algoritmo de visión artificial [**?**], [**?**], que se encarga de extraer características distintivas de las imágenes en escala de grises. Mediante estas, es posible luego reconocer dicha imagen dentro de una base de datos o incluso dentro de otra imagen mayor con otra cantidad de elementos en desorden. Estas características

son invariantes a factores de escala, traslación, rotación y parcialmente invariantes a cambios de iluminación y afinidades. El algoritmo consta básicamente de cuatro pasos que se explicarán brevemente en secciones subsiguientes.

1.4.1. Detección de extremos en el espacio-escala.

Se busca encontrar dentro del *scale-space* (espacio-escala) de la imagen puntos característicos; invariantes a la traslación, el escalado y la rotación de la misma. Además esos puntos deben ser mínimamente afectados por el ruido y pequeñas distorsiones. Serán los puntos extremos (máximos o mínimos) obtenidos de las diferencias Gaussianas aplicadas al *scale-space* de la imagen. El *scale-space* de una imagen se define como una familia de imágenes $L(x,y,\sigma)$ que se obtienen de convolucionar un núcleo Gaussiano variable en su desviación estándar $G(x,y,\sigma)$ con una imagen de entrada $I(x,y)$:

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y)$$

donde $*$ denota la convolución en x e y, y además:

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Una imagen diferencia de Gaussianas, $D(x,y,\sigma)$, se define entonoces de la siguiente manera:

$$D(x,y,\sigma) = L(x,y,k\sigma) - L(x,y,\sigma)$$

con k un factor multiplicativo constante.

Dado un valor inicial para σ, se le realiza a la imagen un número s de diferencias Gaussianas con la desviación estándar variando de manera creciente a lo largo de una octava (hasta obtener un $\sigma' = 2\sigma$). Para obtener s intervalos enteros dentro de dicha octava el valor de k en cada diferencia Gaussiana debe ser de $2^{\frac{1}{s}}$. Una vez calculadas las diferencias Gaussianas a lo largo de la octava, la imagen se submuestrea tomando 1 de cada 2 píxeles en filas y columnas y se procede de la misma manera. La cantidad de octavas involucradas en el cálculo así como la cantidad de diferencias Gaussianas calculadas por octava son un parámetro a determinar. En la figura 1.3 se puede ver lo anterior explicado gráficamente.

Una vez que se obtiene la "pirámide" de diferencia de Gaussianas anterior, se buscan para cada "piso" de la misma extremos locales quienes se transformarán en candidatos a puntos clave. Para una $D(x,y,\sigma)$ determinada y en una octava determinada, un punto (x_0,y_0) será un máximo (mínimo) relativo si es mayor (menor) a sus 8 puntos vecinos dentro de su nivel y a sus 9 puntos vecinos de cada uno de los niveles inferior y superior. Si el punto se encuentra en una $D(x,y,\sigma)$ de transicion entre 2 octavas, se buscan los puntos vecinos correspondientes del nivel superior (inferior). Ver figura 1.4.

1.4.2. Localización exacta de puntos clave.

La búsqueda de extremos en las diferencias de Gaussianas produce múltilpes candidatos entre los que se encuentran puntos con poco contraste; los cuales no son estables a cambios de iluminación y al ruido. Para quitarlos se procede de la siguente manera.

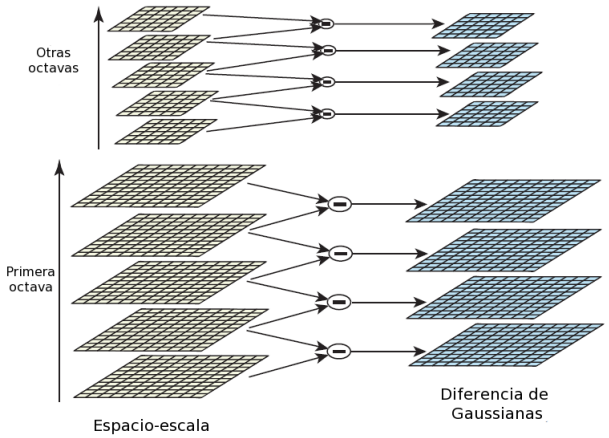

Figura 1.3: Para cada octava, la imagen original es convolucionada repetidamente con Gaussianas de desviación estándar variable para producir el *scale-space* de la izquierda. Imágenes adyacentes del *space-scale* son restadas para lograr la diferencia de Gaussianas de la derecha. Después de cada octava, la imagen borrosa es submuestreada por un factor de dos y el proceso se repite. Figura tomada de [**?**].

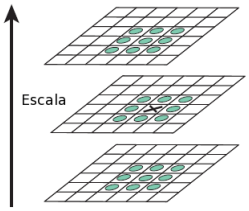

Figura 1.4: Máximos y mínimos de las imágenes diferencia de Gaussianas son obtenidos comparando cada píxeles con sus vecinos en la misma escala, y en las escalas adyacentes. Figura tomada de [**?**].

Primero se realiza una expansión de Taylor de grado 2 entorno a cada extremo detectado (x_0, y_0, σ_0):

$$D(\chi) = D + \frac{\partial D^T}{\partial \chi}\chi + \frac{1}{2}\chi^T \frac{\partial^2 D}{\partial \chi^2}\chi \tag{1.1}$$

donde D y sus derivadas son evaluadas siempre en el punto en cuestión y $\chi = (x, y, \sigma)^T$ es la posición relativa al mismo. Derivando la aproximación anterior e igualándola a cero se obtiene:

$$\bar{\chi} = -\frac{\partial^2 D^{-1}}{\partial \chi^2}\frac{\partial D}{\partial \chi} \tag{1.2}$$

Reemplazando (1.2) en (1.1) se obtiene el valor del máximo local:

$$D(\bar{\chi}) = D + \frac{1}{2}\frac{\partial D^T}{\partial \chi}\bar{\chi}$$

Finalmente, si $|D(\bar{\chi})| < 0{,}03$ el punto es eliminado de la lista de puntos clave; suponiendo que D toma valores entre 0 y 1.

Además de quitar aquellos puntos con poco contraste, hay que quitar a los puntos candidatos que pertenecen a una línea y no a una esquina. Para ello, sea H la matriz Hessiana de $D(x, y, \sigma)$

evaluada en un punto extremo de las diferencias de Gaussianas determinado (x_0, y_0, σ_0), se estará en presencia de un borde (línea) si sus valores propios α y β son uno grande y el otro pequeño. Lo anterior es equivalente a trabajar con los siguientes resultados:

$$Traza(H) = \frac{\partial^2 D}{\partial x^2} + \frac{\partial^2 D}{\partial y^2} = \alpha + \beta$$

$$Det(H) = \frac{\partial^2 D}{\partial x^2} \times \frac{\partial^2 D}{\partial y^2} - \left(\frac{\partial^2 D}{\partial x.\partial y}\right)^2 = \alpha.\beta$$

Sea la $\alpha = r.\beta$, la condición se reduce a:

$$\frac{Traza(H)^2}{Det(H)} < \frac{(r+1)^2}{r}$$

Luego de varios experimentos, se propone un umbral de $r = 10$. Véase que conforme aumenta la relación r entre ambos valores propios también lo hace la relación entre el cuadrado de la traza de la matriz Hessiana y su determinante.

1.4.3. Asignación de orientación.

Mediante la asignación de una orientación a cada punto de la imagen basada en características locales de la misma, los puntos clave pueden ser descriptos relativos a estas orientaciones y de esta manera lograr características invariantes a las rotaciones. Para cada punto clave obtenido $D(x_0, y_0, \sigma_0)$, se busca su imagen borrosa correspondiente en el espacio escala $L(x, y, \sigma_0)$ y se determina el módulo de su gradiente $m(x, y)$ y la fase del mismo $\theta(x, y)$ utilizando diferencias entre píxeles:

$$m(x,y) = \sqrt{(\Delta L_x)^2 + (\Delta L_y)^2}$$

$$m(x,y) = \sqrt{[L(x+1,y) - L(x-1,y)]^2 + [L(x,y+1) - L(x,y-1)]^2}$$

$$\theta(x,y) = tan^{-1}\left(\frac{\Delta L_y}{\Delta L_x}\right)$$

$$\theta(x,y) = tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right)$$

Para determinar de una forma fiel la orientación de cada punto clave, ésta es determinada tomando en cuenta las direcciones de todos los puntos de la imagen dentro de cierto entorno al mismo. Se genera entonces un histograma de direcciones con valores que varían de a 10 grados, ponderado por el módulo del gradiente y una ventana Gaussiana circular centrada en el punto clave, de desviación estándar igual a 1.5 veces el valor del nivel del en cuestión. Cada máximo en el histograma corresponde a la dirección dominante en el gradiente local y será la asignada al punto clave. Si existen en el histograma otros máximos secundarios de valor mayor o igual al 80 % del máximo principal, estos serán utilizados para generar nuevos puntos clave con esa dirección. Sólo al 15 % de los puntos clave se les asigna más de una dirección.

1.4.4. Descriptor de puntos clave.

Hasta el momento, se le ha asignado a cada punto clave una escala, una locación y una orientación. El siguiente paso es determinar para cada punto clave un descriptor relativamente invariante a cambios de iluminación y afinidades, basado en el entorno del mismo.

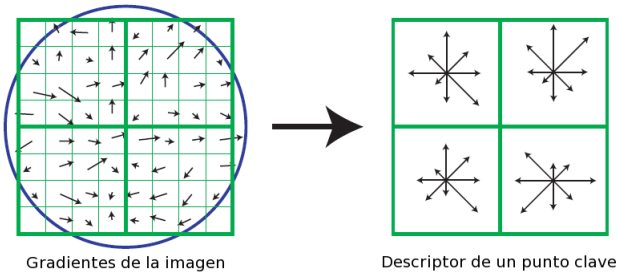

Figura 1.5: Izq.: La ventana Gaussiana pondera los valores de módulo y fase en la vecindad de los puntos de interés. Der.: los histogramas con 8 direcciones posibles realizados para cada subregión. Figura tomada de [**?**].

Una vez determinadas la magnitud y fase del gradiente entorno a un punto clave, una ventana Gaussiana centrada en este pondera los valores de módulo y fase de 4×4 subregiones cuadradas en la vecindad del mismo, cada una formada por 16 píxeles. Nuevamente se genera para cada subregión un histograma de 8 direcciones distintas. Se obtiene finalmente para cada punto clave un descriptor de $4 \times 4 \times 8 = 128$ valores.

En la figura 1.5 se ve cómo se computan los descriptores para cada punto clave. En el ejemplo se utilizan únicamente $2 \times 2 = 4$ subregiones en vez de $4 \times 4 = 16$.

[**?**].