

UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE CUENCA



FACULTAD DE INGENIERIAS
CARRERA DE SISTEMAS

TESIS PREVIA LA OBTENCIÓN DEL TÍTULO DE
INGENIERO DE SISTEMAS

TÍTULO :
RECUPERACIÓN DE IMÁGENES BASADA EN
CONTENIDOS UTILIZANDO EL MÉTODO SURF

Autor:
Tania Jacqueline Patiño Ordoñez

Director :
Ing. Vladimir Robles B.

Cuenca, Febrero 2012

Tania Jacqueline Patiño Ordoñez: *Recuperación de imágenes basada en contenidos utilizando el método SURF*, © Febrero 2012

Ing. Vladimir Robles B.

CERTIFICA

Haber dirigido y revisado prolijamente cada uno de los capítulos del informe de tesis, realizado por la Sra. Tania Jacqueline Patiño Ordoñez, y por cumplir los requisitos autorizo su presentación.

Cuenca, Febrero 2012

Ing. Vladimir Robles B.
DIRECTOR DE TESIS

DECLARACIÓN DE RESPONSABILIDAD

Los conceptos, análisis, conclusiones y recomendaciones desarrollados en la presente tesis, son de exclusiva responsabilidad del autor. Los textos de otros autores llevan su correspondiente cita bibliográfica.

Cuenca, Febrero 2012

Tania Jacqueline Patiño Ordoñez

*No somos nadie sin la ayuda de Dios, gracias a
Él tenemos todo lo que poseemos, sé agradecido
con todas las personas que nunca sabes qué
rumbo toman los caminos de la vida...*

AGRADECIMIENTOS

Quiero agradecer primeramente a Dios por darme la dicha de la vida y dejarme disfrutar día a día de mi familia.

A mi papá Ángel Patiño quien desde niña me brindó todo su esfuerzo y dedicación para cumplir con todos mis sueños.

A mi madre María Ordóñez quien, como toda madre siempre se preocupa de mí y me brinda sus consejos para ser una mejor persona buscando el porvenir de cada uno de sus hijos.

A una persona muy especial, mi amado esposo Miguel Chacha quien con su amor y a pesar de la distancia siempre me estuvo apoyando e impulsando para que culmine mis estudios.

A mi mas grande tesoro, mi hijo Jacob Sebastian por quien vivo y por quien hasta soy capaz de morir, quien con su carita angelical y dulce, siendo tan solo un niño me supo entender y apoyar en todo momento.

A mi hermano Marco Paúl que aunque esté lejos, siempre estuvo dándome una palabra de aliento para no dejarme vencer por mis miedos.

A mis tíos y primos que a pesar de la distancia estuvieron pendiente de mí.

A mis amigos Alba y Pedro con quienes estreché lazos de amistad muy grandes e hicieron que mis días en el laboratorio sean llenos de alegría.

A mis amigas y amigos María Paredes, Marjoury Matamoros, Johanna León, Fernanda Sanchez, Carlos Lucero, Diego Lozada, Patricio Luna, quienes hicieron que mi paso por la Universidad sean momentos gratificantes e inolvidables recuerdos.

Un agradecimiento especial a mi profesor y director de tesis Ing. Vladimir Robles quien con su constante e incansable ayuda hizo que esta tesis se lleve a cabo, brindándome siempre una mano amiga en todo momento.

Muchas gracias de todo corazón, siempre estaré agradecida con Dios y la vida por haberme puesto en el camino a personas como ustedes.

Tania Jacqueline Patiño Ordoñez

DEDICATORIA

Esta tesis va dedicada primeramente a Dios, por guiarme siempre con su luz divina y a todas las personas importantes en mi vida que de una u otra forma estuvieron apoyandome para llegar a cumplir una de mis metas que ya no quedó expresado solo como un sueño, sino que hoy se plasma en realidad. Entre esas personas importantes quiero dedicar a:

Mis padres Angel y Maria, por ser los
pilares fundamentales en mi vida.

Mi esposo Miguel Chacha, por su
amor y apoyo constante.

Mi hijo Jacob Sebastian, por ser
ser la razón que me impulsa
a seguir adelante

A mis amigos quienes me daban palabras
de aliento para no dejarme caer.

Tania Jacqueline Patiño Ordoñez

ÍNDICE GENERAL

I MÉTODO SIFT 5

1	SIFT	7
1.1	Introducción	7
1.2	Fundamento teórico	7
1.2.1	Detección de extremos espacio-escala	7
1.2.2	Localización de los puntos claves	8
1.2.3	Asignación de orientación	8
1.2.4	Descriptor de puntos claves:	9
1.3	Aplicaciones	9
1.3.1	Reconocimiento de Objetos	9
1.3.2	Reconocimientos de rostros	10
1.3.3	Navegación de robots	10
1.3.4	Construcción de panoramas	10
1.4	Variantes SIFT	10
1.4.1	ASIFT (Affine Scale Invariant Feature Transform)	10
1.4.2	PCA-SIFT (Principal Components Analysis-SIFT)	12
1.4.3	GLOH (Gradient Location-Orientation Histogram)	14
1.5	Herramientas disponibles	15
1.5.1	FPGA	15
1.5.2	SIFTGPU	15

II MÉTODO SURF 17

2	SURF	19
2.1	Introducción	19
2.2	Fundamento teórico	19
2.2.1	Detección de puntos de interés	20
2.2.2	Asignación de Orientación	22
2.2.3	Creación del descriptor	22
2.3	Aplicaciones	24
2.3.1	Reconocimiento de imágenes móviles	24
2.3.2	Reconocimiento de rostros	24
2.3.3	Estimación de distancia recorrida por medio de robot	24
2.4	Variantes SURF	25
2.4.1	SURF-128	25
2.4.2	U-SURF (Upright-SURF)	25
2.5	Herramientas disponibles	25
2.5.1	KOOABA	25
2.5.2	Open CV	25
2.5.3	CUDA SURF	25

III WAVELETS 27

3	WAVELETS	29
3.1	Introducción	29
3.2	Fundamento teórico	30
3.2.1	Transformada Wavelet Continua	30
3.2.2	Transformada Wavelet Discreta(DWT)	32
3.3	Aplicaciones	36
3.3.1	Análisis de señales	36

3.3.2	Aplicación al diseño textil	36
3.3.3	Reconocimiento de rostros	36
3.3.4	Extracción de características en la identificación de Patologías de ECG	37
3.4	Variantes Wavelets	37
3.4.1	Wavelets Haar	37
3.4.2	Daubechies	38
3.4.3	Gabor	38
3.5	Herramientas disponibles	38
3.5.1	Scilab wavelet toolbox	38
3.5.2	Wavelab	39
3.5.3	LabView Add-On: Procesamiento avanzado de señales	40
IV SELECCIÓN DE LA HERRAMIENTA DE EXPERIMENTACIÓN		
41		
4	SELECCIÓN DE LA HERRAMIENTA DE EXPERIMENTACIÓN	43
4.1	Revisión de las herramientas disponibles	43
4.1.1	Método Original	43
4.1.2	OpenSurf	43
4.1.3	OpenSurfCL	43
4.1.4	OpenCV SURF	43
4.1.5	SurfMex	44
4.1.6	OpenSURF Matlab	44
4.1.7	Libmv SURF	44
4.1.8	JOpenSURF	44
4.1.9	Parallel SURF	44
4.1.10	FPGA SURF	44
4.2	Comparativa de las herramientas	45
4.3	Selección y pruebas de la herramienta	47
V EXPERIMENTACIÓN		
49		
5	EXPERIMENTACIÓN	51
5.1	Selección del corpus	51
5.2	Diseño del Plan de Experimentación	52
5.3	Ejecución y Resultados	53
5.4	Análisis de Resultados	57
BIBLIOGRAFÍA		
65		
VI ANEXOS		
69		
A	ANEXOS	71
A.1	Codigo SURF	71

ÍNDICE DE FIGURAS

Figura 1	Comparación entre píxeles vecinos .	8
Figura 2	Descriptor de puntos claves formado por 128 gradientes [12].	9
Figura 3	Resultado al aplicar SIFT a la imagen [28]	9
Figura 4	Perspectiva de la cámara según el ángulo (el piso se convierte en rectángulo, las baldosas en paralelogramos)[16]	11
Figura 5	Inclinaciones de la imagen utilizando una t direccional [16]	11
Figura 6	Comparativa de los métodos ASIFT, SIFT, MSER en la obtención de los matchs	12
Figura 7	Gradientes horizontal y vertical de acuerdo a la orientación dominante[3]	13
Figura 8	Conjunto de imágenes de entrada para el proceso de PCA	13
Figura 9	Resultado de imágenes medias realizadas al conjunto de imágenes originales	14
Figura 10	División de regiones en el método GLOH [25]	15
Figura 11	Imagen submuestreada	19
Figura 12	Técnica espacio-escala	20
Figura 13	Aproximaciones de las derivadas parciales usando filtros tipo caja[10]	21
Figura 14	Imagen Integral, utilizando cuadrados para promediar la zona de la imagen [27]	21
Figura 15	Convolución del filtro Haar con una imagen integral dividida en 4 sub-regiones, donde 1,2,3 y 4 son los resultados de la imagen integral[6]	22
Figura 16	Asignación de orientación [10]	22
Figura 17	Imagen con las regiones cuadradas centradas alrededor de los puntos de interés de tamaño 20s [10]	23
Figura 18	Respuestas Haar de tamaño de 2s, la parte de color negro representa un peso de -1 y el blanco 1[10]	23
Figura 19	Respuestas de Haar en cada una de las Sub-regiones alrededor del punto de interés	23
Figura 20	Resultado aplicando el método SURF[28]	24
Figura 21	Función Wavelet con sus escalaciones y traslaciones correspondientes, imagen tomada de [11]	30
Figura 22	Tipos de coeficientes al aplicar la DWT [26]	32
Figura 23	Localización de las wavelets discretas en tiempo-escala con muestreo diádico, tomada de [23]	33
Figura 24	Expansión y desplazamiento en factor 2, imagen tomada de [23]	34
Figura 25	Espectros enlazados secuencialmente. [23]	34
Figura 26	Función escalado, imagen tomada de [23]	35
Figura 27	Función Haar	37
Figura 28	Wavelet Daubechies de orden 4 tomada de [11]	38
Figura 29	Wavelets Gabor en el reconocimiento de rostros	39

Figure 30	Pruebas de la herramienta con la imagen por defecto	48
Figure 31	Pruebas de la herramienta con la segunda imagen	48
Figura 32	Ejemplos de PPM binario y ASCII	51
Figura 33	Imágenes del Corel Data Set	52
Figura 34	Diagrama del plan de experimentación (los rectángulos ovalados representan procesos y los rectángulos comunes datos)	53
Figura 35	Operaciones aplicadas a la Imagen	54
Figura 36	Detección de puntos de correspondencia aplicando SURF	56
Figura 37	Utilizando las 5 imágenes mejor valoradas según los puntos de coincidencias	58
Figure 38	Análisis de precisión del experimento	59

ÍNDICE DE CUADROS

Cuadro 1	Tabla comparativa de las herramientas SURF.	46
Cuadro 2	Tiempo de duración en el proceso de obtención de puntos descriptores	55
Cuadro 3	Resultados de precisión de acuerdo al numero de imágenes mejor valoradas .	57

INTRODUCCIÓN

En la actualidad, con la creciente demanda de información se ve la necesidad de recurrir a métodos mucho más rápidos y eficaces para la recuperación de la misma, por lo que es primordial contar con bases de datos de gran capacidad no solo para almacenar información sino también para imágenes que requieren de mucho más espacio y sin pérdida de ninguna de sus características. Una vez almacenada las imágenes en la base de datos es necesario la recuperación de dichas imágenes que serán procesadas y tratadas utilizando alguna técnica o método específico. En nuestro caso vamos a utilizar imágenes de entrenamiento para compararlas con sus variantes rotadas, trasladadas y escaladas localizando puntos de interés o rasgos sobresalientes entre las imágenes, siendo estos puntos invariantes a las operaciones de rotación, traslación y escala.

La recuperación de imágenes basada en contenidos (CBIR), se fundamenta en comparar una imagen "X" con otra imagen "Y" e identificar sus similitudes, es muy utilizado en muchos ámbitos como la medicina, registros policiales, detección de rostros, etc¹.

En CBIR existen diferentes técnicas de comparación basada en contenidos [5] como son:

- Color.- Implica comparar el histograma de color, siendo esta técnica la más utilizada puesto que no depende del tamaño o de su orientación.
- Forma.- Las detección de formas se realiza por segmentación o localización de bordes.
- Textura.- Busca patrones visuales y cómo se definen en el espacio.

Existen muchos métodos que utilizan CBIR como son SIFT, SURF, Wavelets siendo estos motivo de nuestro estudio. El objetivo principal es realizar un experimento para comprobar la efectividad del método SURF, pero para ello se realiza un estudio de los métodos mas utilizados en la actualidad. El documento esta conformado de la siguiente manera: En los capítulos 1, 2 y 3 se estudia el proceso que realizan los métodos SIFT, SURF, Wavelets, así como sus variantes, aplicaciones y herramientas disponibles, respectivamente; en el capítulo 4 se hace una revisión de las herramientas disponibles que utiliza SURF así como de los otros dos métodos (SIFT y wavelets), a partir de esto se selecciona una para emplear en el experimento; en el capítulo 5 se lleva a cabo el proceso de experimentación, en el cual se detalla las imágenes utilizadas, el plan de experimentación, y se realiza un análisis de los resultados obtenidos al aplicar el método SURF; por ultimo se anexa el código de SURF utilizado en el experimento, se realiza las conclusiones y recomendaciones basados en el experimento y en el documento en general.

¹ Fuente: http://es.wikipedia.org/wiki/Consulta_de_im%C3%A1genes_mediante_ejemplo, consultado el: 25/10/2011

OBJETIVOS

GENERAL

- Conocer y estudiar las técnicas de recuperación basada en contenidos centrándose en el método SURF

ESPECIFICOS

- Realizar un estudio sobre la recuperación de imágenes basada en contenidos.
- Realizar una comparativa entre los métodos como SIFT, SURF, Wavelets, así como sus aplicaciones.
- Investigar las herramientas disponibles para implementar aplicaciones basadas en técnicas antes mencionadas.
- Seleccionar una herramienta para implementar la técnica SURF y realizar pruebas de funcionamiento y evaluación
- Establecer un plan de experimentación para probar su funcionamiento y rendimiento de la técnica SURF.
- Revisar y preparar un corpus de imágenes a fin de realizar pruebas de funcionamiento de la técnica SURF

Part I

MÉTODO SIFT

SIFT

1.1 INTRODUCCIÓN

El método SIFT (Scale Invariant Feature Transform) fue descrito por David G. Lowe [19] en el año 2004, su nombre se debe a que permite detectar puntos de interés de una imagen invariantes a la escala, rotación, cambio de iluminación y al punto de vista en 3D [1].

El método SIFT en la actualidad está siendo muy difundido puesto que es utilizado en muchas aplicaciones. Se plantea una imagen con un objeto en una determinada posición y al tener una nueva imagen con el mismo objeto pero rotado o escalado, con SIFT reconoce el objeto lo cual es invariante a todos estos cambios gracias a las propiedades que utiliza este algoritmo.

SIFT utiliza la diferencia de Gauss para suavizar la imagen quedando así solo los puntos de interés.

1.2 FUNDAMENTO TEÓRICO

Este método se basa en detectar los puntos relevantes de una imagen, los mismos que son invariantes a la escala, rotación y traslación. Consta de los siguientes pasos:

1.2.1 Detección de extremos espacio-escala

Para la detección de los extremos se utiliza el filtro Gaussiano (ecuación 1.1), puesto que es invariante a la escala eliminando de esta manera el ruido de la imagen.

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \quad (1.1)$$

La imagen con la que se procede a trabajar es el resultado de una convolución $L(x, y, \sigma)$ entre la imagen original $I(x, y)$ y la que se realizó el filtro Gaussiano $G(x, y, \sigma)$, quedando de esta manera la ecuación, tal como plantea el autor en [4]:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (1.2)$$

donde σ representa la escala, también se hace una diferencia de dos filtros gaussianos multiplicados por una constante k y luego se convoluciona con la imagen original para el suavizado de dicha imagen, como se muestra en la ecuación tomada de [1]:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \quad (1.3)$$

Este algoritmo plantea una diferencia Gaussiana (DoG) de cada octava, ver ecuación 1.4:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \quad (1.4)$$

Octava: Conjunto de imágenes suavizadas con imágenes DoG.

1.2.2 Localización de los puntos claves

Al tener los puntos de interés determinados con el paso anterior, se estudian los píxeles y se comparan con los píxeles vecinos, solo se quedan los puntos que son estables, los otros puntos de bajo contraste y los que están en los bordes son eliminados como se observa en la figura 1.

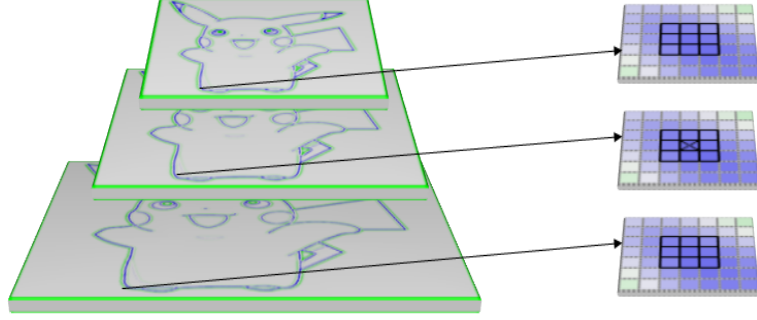


Figura 1: Comparación entre píxeles vecinos .

Para la eliminación de los puntos de bajo contraste se utiliza un proceso de umbralización (D) en la cual si los puntos son menores a D serán eliminados, siendo $D=0,03$ según plantea Lowe. Los puntos que se encuentran en los bordes son eliminados a través de la función DoG por medio de la matriz Hessiana, tomado de [4]:

$$H = \begin{bmatrix} \frac{\partial^2 D}{\partial x^2} & \frac{\partial^2 D}{\partial x \partial y} \\ \frac{\partial^2 D}{\partial x \partial y} & \frac{\partial^2 D}{\partial y^2} \end{bmatrix} \quad (1.5)$$

siendo D la imagen DoG, con la siguiente desigualdad se localiza los puntos en los bordes y los que no cumplan dicha desigualdad serán descartados, con $r = 10$, selección heurística que se realiza el autor [19]:

$$\frac{\left(\frac{\partial^2 D}{\partial x^2} + \frac{\partial^2 D}{\partial y^2} \right)^2}{\left(\frac{\partial^2 D}{\partial x^2} \times \frac{\partial^2 D}{\partial y^2} \right) - \left(\frac{\partial^2 D}{\partial x \partial y} \right)^2} < \frac{(r+1)^2}{r} \quad (1.6)$$

1.2.3 Asignación de orientación

A cada punto clave se asigna una o mas orientaciones con respecto a las direcciones del gradiente, para ello se calcula la magnitud del gradiente (m) y la orientación θ , por medio de la siguiente ecuación, tomada de [8].

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \quad (1.7)$$

$$\theta(x, y) = \arctan \frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)} \quad (1.8)$$

A partir de las orientaciones de cada punto clave se crea un histograma de 36 bins, el bin con valor más alto de magnitud es considerado como orientación dominante del gradiente.

1.2.4 Descriptor de puntos claves:

Se forma un vector para cada punto clave, primero se muestrean las orientaciones y magnitudes alrededor de cada punto sobre regiones de 16×16 , resumiendo en sub-regiones de 4×4 cada uno con 8 bins, por lo tanto un descriptor esta formado por 128 elementos o gradientes como se muestra en la figura 2:

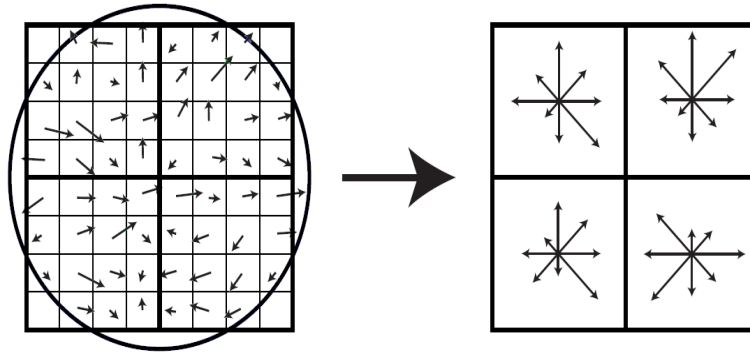


Figura 2: Descriptor de puntos claves formado por 128 gradientes [12].

En la figura 3 se muestran los resultados después de aplicar el método SIFT

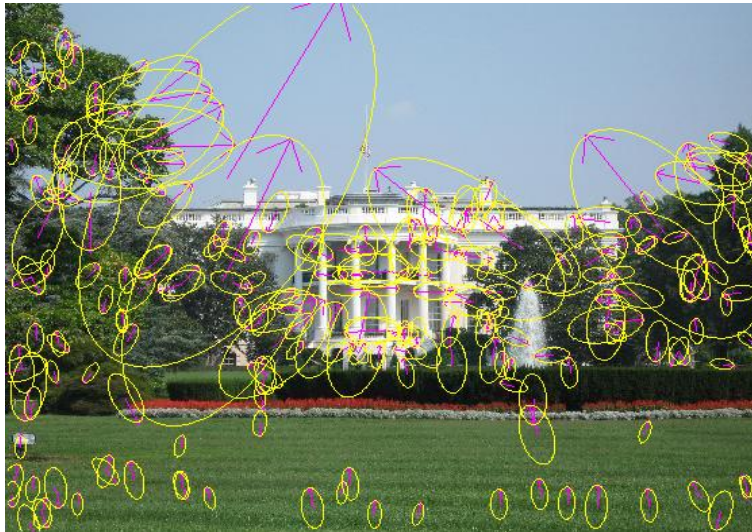


Figura 3: Resultado al aplicar SIFT a la imagen [28]

1.3 APLICACIONES

1.3.1 Reconocimiento de Objetos

Consiste en extraer los puntos principales de una imagen, estos puntos son invariantes a la escala, rotación, traslación, analiza las imágenes

utilizando filtros Gaussianos, matrices Hessianas descritas en la sección de Fundamento Teórico .

1.3.2 Reconocimientos de rostros

SIFT es invariante a la escala, rotación, cambios de iluminación y traslación por lo que es empleando para la detección de rostros, pudiendo la imagen sufrir cambios y sin embargo logra detectar los rostros de manera eficiente. Tras pruebas realizadas por autores sobre imágenes de rostros, se citan las siguientes ventajas de invariancia con respecto a los rasgos obtenidos [13]:

- A la escala.
- Rotación
- Traslación..
- Cambios de iluminación.

1.3.3 Navegación de robots

Por medio de una cámara integrada que lleva el robot, se captura imágenes las cuales serán procesadas utilizando el método SIFT extrayendo puntos claves, estos puntos son tomados de referencia para estimar la posición y el movimiento, y luego se analiza si los puntos conciden entre dos imágenes.

1.3.4 Construcción de panoramas

Se usan características invariantes que permiten detectar imágenes panorámicas rotadas, escaladas o con cambios de iluminación, con SIFT se puede detectar panoramas sin importar que los datos estén desordenados y este es capaz de reconocer con precisión los datos presentados en las imágenes.

1.4 VARIANTES SIFT

1.4.1 ASIFT (*Affine Scale Invariant Feature Transform*)

Mediante ASIFT se pueden identificar con mayor exactitud las características de una imagen que hayan sido distorsionadas por un nuevo parámetro como la inclinación. Según [37], la desventaja de ASIFT es que exige un costo computacional elevado, siendo hasta 12 veces mayor que el método SIFT.

ASIFT es totalmente invariante, ya que simula dos parámetros del eje de la cámara (latitud y longitud) y luego aplica el método SIFT con el cual normaliza la rotación y traslación [16].

A continuación se explica la forma en que opera el método ASIFT:

- El ángulo de captura que realiza la cámara, hará que la perspectiva de la imagen cambie, es decir, como se observa en la figura 4, el piso se convierte en un rectángulo y a su vez cada baldosa es un paralelogramo y este proceso es conocido como distorsiones.



Figura 4: Perspectiva de la cámara según el ángulo (el piso se convierte en rectángulo, las baldosas en paralelogramos)[16]

- La imagen es transformada a todas las posibles distorsiones producidas por la posición de la cámara (latitud θ y longitud ϕ), debiendo ser sometida a una rotación ϕ y luego a una inclinación de $t = \left| \frac{1}{\cos \theta} \right|$. Para el caso de imágenes digitales, la inclinación se realiza por un t direccional como se muestra en la figura 5, por lo tanto requiere un filtro de suavizado ó una convolución gaussiana con una desviación estándar $c\sqrt{t^2 - 1}$ con $c = 0,8$, valor elegido heurísticamente por Lowe [19] en el método SIFT.

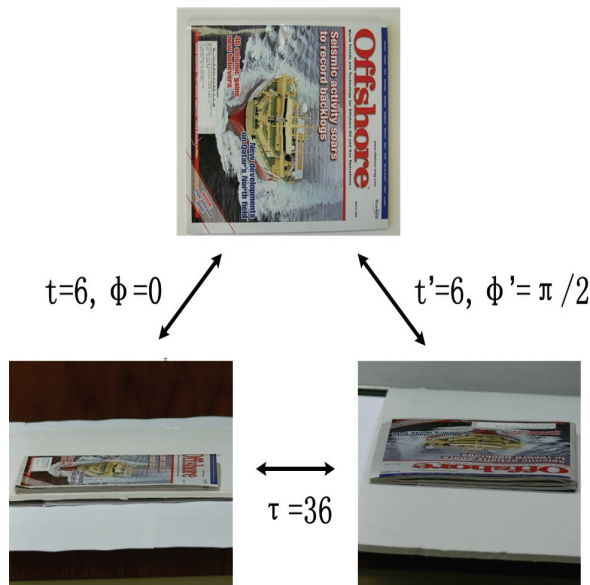
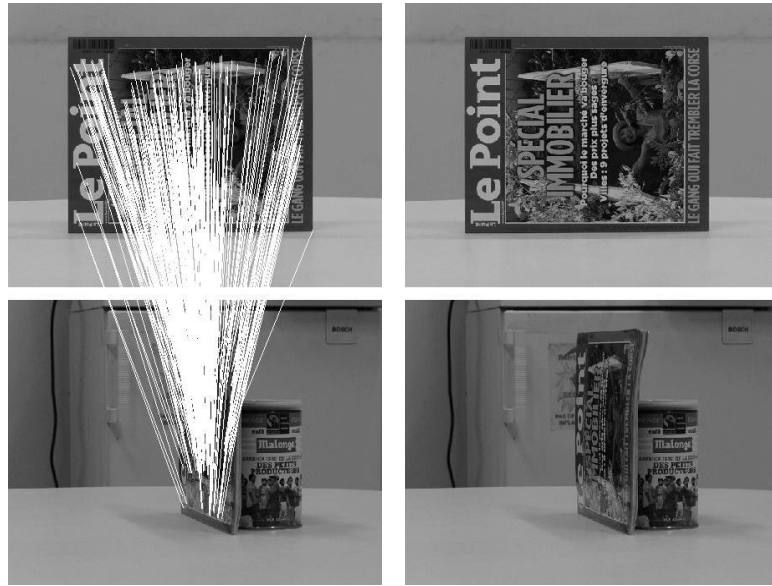


Figura 5: Inclinaciones de la imagen utilizando una t direccional [16]

- Se realiza el proceso de comparación entre imágenes, para ello utiliza el SIFT.

En la figura 6¹ se demuestra la efectividad resultante entre los métodos ASIFT, SIFT, MSER, se puede observar que ASIFT presenta un mayor manejo de transición en la inclinación obteniendo una mayor cantidad de puntos claves.

¹ Fuente: <http://www.cmap.polytechnique.fr/~yu/research/ASIFT/demo.html>, consultado el: 25/01/2012



(a) Método ASIFT con 349 matches

(b) Método SIFT con 0 matches



(c) Método MSER con 17 matches

Figura 6: Comparativa de los métodos ASIFT, SIFT, MSER en la obtención de los matches

1.4.2 PCA-SIFT (Principal Components Analysis-SIFT)

PCA-SIFT permite reducir la dimensión de los descriptores lo cual hace que sea mucho mas robusto frente a cambios de iluminación, que su predecesor SIFT

PCA-SIFT al igual que SIFT determina la posición, escala y orientación de los puntos de interés y realiza un análisis de gradiente en una región de 41×41 , selección heurística que realiza el autor en [36], luego voltea la ventana de acuerdo a la orientación dominante para formar un vector que concatena los gradientes horizontal y vertical

como se muestra en la figura 7. La dimensión de los vectores obtenidos es $2 \times 39 \times 39 = 3042$ según [25].

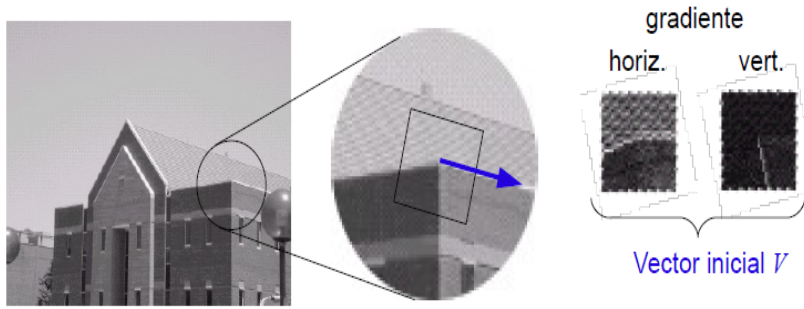
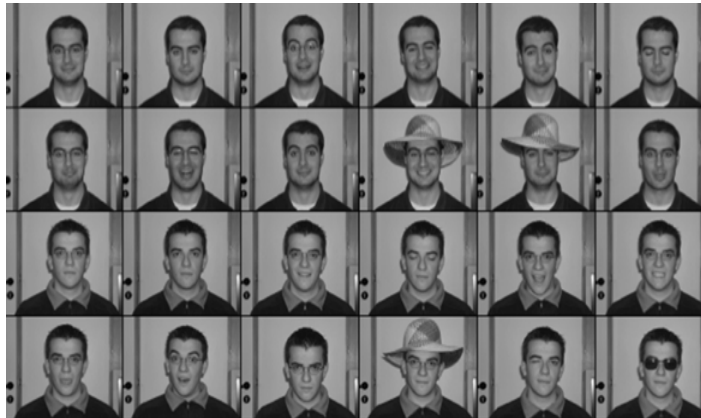


Figura 7: Gradientes horizontal y vertical de acuerdo a la orientación dominante[3]

Posteriormente se debe reducir la dimensión de los vectores para lograr una mayor singularidad, para ello utiliza PCA, a continuación se dará una breve explicación del proceso que realiza PCA[?] empleado en EigenFaces:

- Recibe un conjunto de entrenamiento $C = \{x_i\}, i = 1 \dots P$, en este caso la imagen, como se muestra en la figura 8².



EigenFaces: Son conjunto de vectores utilizados para captar las variaciones de una colección de imágenes de rostros

Figura 8: Conjunto de imágenes de entrada para el proceso de PCA

- Cada elemento es un vector $x_i = [x_{i_1}, x_{i_2}, \dots, x_{i_n}]^T$
- Calcular la media de la imagen $\bar{x} = \frac{1}{P} \sum x_i$.
- Centrar el conjunto de entrenamiento en el origen, para ello se resta la media a cada imagen con $x'_i = x_i - \bar{x}$
- Construir una matriz, siendo cada columna un elemento del conjunto de entrenamiento

$$X = \begin{bmatrix} x'_{1_1} & x'_{2_1} & \dots & x'_{P_1} \\ x'_{1_2} & x'_{2_2} & \dots & x'_{P_2} \\ \vdots & \vdots & \ddots & \vdots \\ x'_{1_N} & x'_{2_N} & \dots & x'_{P_N} \end{bmatrix}$$

² Fuente: <http://rua.ua.es/dspace/bitstream/10045/17323/7/reconocimiento.pdf>, consultado el: 24/01/2012,

- Calcular la matriz de covarianza $Q = XX^T = \frac{1}{N} \sum_{i=1}^N x_i' x_i'^T$
- Calcular los autovectores v_l, u_l

$$u_l = \sum_{k=1}^M v_{lk} x_k \quad l = 1, \dots, M$$

El resultado va a ser una imagen media y las correspondientes para cada autovector como se presenta en la figura 9.

3



Figura 9: Resultado de imágenes medias realizadas al conjunto de imágenes originales

PCA-SIFT al igual que SIFT utiliza la distancia euclídea para determinar la correspondencia entre los puntos claves de dos imágenes.

1.4.3 GLOH (Gradient Location-Orientation Histogram)

Opera de manera similar a SIFT, fue diseñado para ser más robusto en los cálculos de los descriptores, presenta las siguientes diferencias con respecto a SIFT:

- Los cálculos de los descriptores son realizados mediante divisiones de regiones circulares en el sistema de coordenadas polares, creando así histogramas para la orientación de los puntos claves.
- En la figura 10 se puede observar que posee tres contenedores en dirección radial y ocho en dirección angular [25].
- La orientación del gradiente es de 16 bins, a diferencia de SIFT, que tiene una orientación del gradiente con 32 bins
- Posteriormente a la obtención de los descriptores, se debe reducir la dimensionalidad de los mismos para ello utiliza PCA, explicado anteriormente en la sección 1.4.2.

3 Fuente: <http://rua.ua.es/dspace/bitstream/10045/17323/7/reconocimiento.pdf>, consultado el: 24/01/2012,

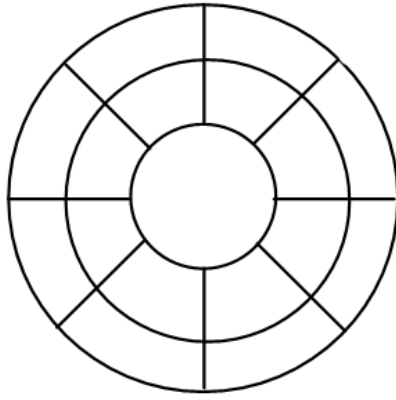


Figura 10: División de regiones en el método GLOH [25]

1.5 HERRAMIENTAS DISPONIBLES

1.5.1 *FPGA*

Es un dispositivo programable por el usuario, está compuesto por bloques lógicos que se comunican por conexiones programables según cita el autor en [34]. FPGA es utilizado para acelerar procesos en reconocimiento de rostros utilizando para ello SIFT que interactúa de forma paralela en el procesamiento. En [33] el autor afirma que utiliza un sistema que implementa SIFT en FPGA para ayudar a la navegación de un robot, realiza la comparación de usar un procesador Pentium III de 700MHz que tardaría 600ms el sistema en procesar una imagen mientras que al utilizar FPGA tarda 60 ms, lo cual es muy notable la reducción de tiempo.

1.5.2 *SIFTGPU*

Esta es una implementación de SIFT usando en GPU (Unidad de Procesamiento Gráfico). Usa procesos en paralelo para la creación de pirámides de Gauss que son utilizadas en el método SIFT para la obtención de los puntos así como la orientación de los mismos y los descriptores correspondientes.

Part II

MÉTODO SURF

SURF

2.1 INTRODUCCIÓN

El algoritmo SURF (Speeded Up Robust Features), fue planteado por Herbert Bay en el año 2006 [28], se basa en SIFT pero a diferencia de este, es más robusto puesto que utiliza wavelets y determinantes Hessianas, además SURF está libre de patentes y existen numerosas implementaciones mucho más eficientes que SIFT.

Hace réplicas de la imagen para así buscar los puntos que estén en todas las réplicas asegurando la invariancia de escala. Las réplicas pueden ser piramidales o del mismo tamaño a la original. En las piramidales la principal idea es tener una imagen más pequeña que la anterior, para ello se realiza un filtro de la imagen y se submuestra ya que si se realiza solo el submuestreo se pierde la información de la imagen como se muestra en la figura 11.

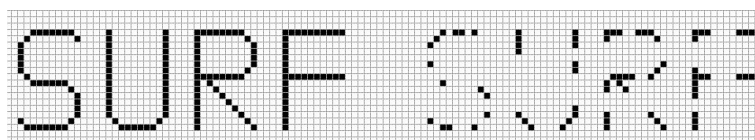


Figura 11: Imagen submuestreada

SURF ofrece la detección de objetos con menor coste computacional, es más rápido y eficaz que SIFT en lo que se refiere a realizar cálculos para la generación de puntos claves, todo esto gracias a la reducción de la dimensionalidad y dificultad en el cálculo de vectores [4].

Al igual que el método SIFT este utiliza la técnica de Scale-Space $L(x, y, \sigma)$ para reducir el ancho de banda de la imagen original [27], a medida que σ va incrementando la imagen se hace más borrosa, ver figura 12¹.

2.2 FUNDAMENTO TEÓRICO

Como ya se dijo anteriormente el método SURF está basado en SIFT, siendo este más robusto en cuanto a cálculos. Utiliza determinantes Hessianas para la detección de los puntos de interés localizando a la vez su posición así como su escala, también utiliza filtros tipo caja para aproximar las derivadas parciales de segundo orden del filtro de Gauss (ecuación 2.2) y ser evaluados de forma mucho más rápida usando imágenes integrales. El análisis espacio-escala se basa en la elevación del tamaño del filtro[4] a diferencia de SIFT que reduce el tamaño de la imagen original para este análisis.

Imágenes Integrales:
Es una representación de la imagen que permite un cálculo rápido de características, debido a que trabaja directamente con las intensidades de la imagen original [22]

¹ Fuente: https://www.u-cursos.cl/ingenieria/2011/1/EL7008/1/rial_docente/bajar?id_material=364752, consultado el: 05/08/2011

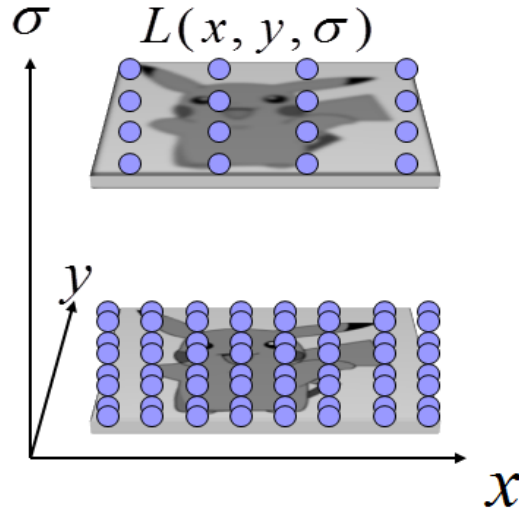


Figura 12: Técnica espacio-escala

2.2.1 Detección de puntos de interés

SURF utiliza la determinante Hessiana para la localización de la posición y escala de los puntos de interés. La matriz Hessiana $H(x, \sigma)$ permite mayor precisión y rapidez en los cálculos ver ecuación 2.1:

$$H(x, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix} \quad (2.1)$$

teniendo en cuenta que la derivada de segundo orden del filtro de Gauss es:

$$\frac{\partial^2}{\partial x^2} g(\sigma) \quad (2.2)$$

se realiza una convolución con la imagen I en el punto $x = (x, y)$ para obtener el valor de $L_{xx}(x, \sigma)$ ver ecuaciones 2.3 y 2.4, de igual manera para obtener $L_{xy}(x, \sigma)$ y $L_{yy}(x, \sigma)$ el proceso es similar, estas derivadas son conocidas como derivadas Laplacianas de Gaussianas.

$$\begin{aligned} L_{xx}(x, y, \sigma) &= \frac{\partial^2}{\partial x^2} * (G(x, y, \sigma) * I(x, y)) \\ &= \left(\frac{\partial^2}{\partial x^2} * (G(x, y, \sigma)) \right) * I(x, y) \end{aligned} \quad (2.3)$$

$$L_{xx}(x, y, \sigma) = G_{xx}(x, y, \sigma) * I(x, y) \quad (2.4)$$

El cálculo de los determinantes de la matriz Hessiana que permite localizar la escala del punto, se define a continuación:

$$\det(H_{\text{aprox}}) = D_{xx}D_{yy} - (\omega D_{xy})^2$$

donde D_{xx} , D_{yy} , D_{xy} son las aproximaciones de las derivadas parciales, ω es el peso para balancear la determinante Hessiana, el calculo para D_{xx} se realiza a partir de $I(x,y)$ y $G_{xx}(x,y)$, de manera similar para D_{yy} , D_{xy} .

El calculo del peso ω se presenta en la ecuación 2.5, tomando en cuenta la figura 13 que utiliza filtros tipo caja de 9×9 con aproximaciones Gaussianas de segundo orden con $\sigma = 1,2$, sugerido en [10], donde $|x|_F$ es la Norma de Frobenius².

$$\omega = \frac{|L_{xy}(1,2)|_F |D_{yy}(9)|_F}{|L_{yy}(1,2)|_F |D_{xy}(9)|_F} = 0,912.. \simeq 0,9 \quad (2.5)$$

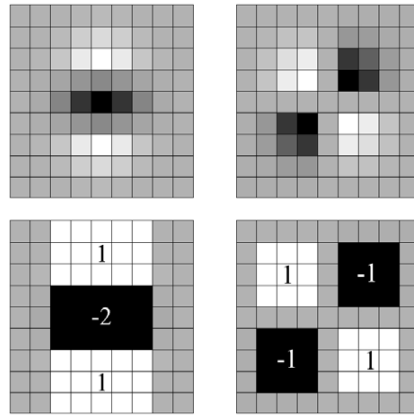


Figura 13: Aproximaciones de las derivadas parciales usando filtros tipo caja[10]

La ventaja de SURF es que utiliza cuadrados (figura13) para la promediar la zona de la imagen, utilizando para ello la Imagen integral (figura 14) descrita en la ecuación 2.6

$$S(x,y) = \sum_{x' \leq x} \sum_{y' \leq y} I(x',y') \quad (2.6)$$

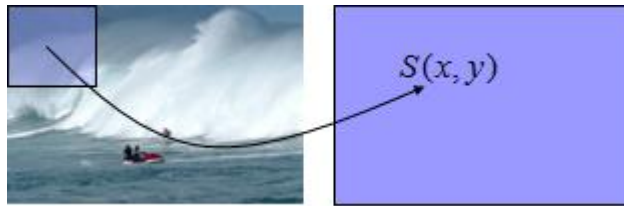


Figura 14: Imagen Integral, utilizando cuadrados para promediar la zona de la imagen [27]

siendo $S(x,y)$ la imagen integral y $I(x',y')$ la imagen original, luego se procede a realizar una convolución entre la imagen integral y el filtro Haar para encontrar la suma ($\text{Sum} = (1 + 4) - (2 + 3)$) del área como se muestra en la figura 15.

² Fuente:http://es.wikipedia.org/wiki/Norma_matricial, consultado el: 06/08/2011

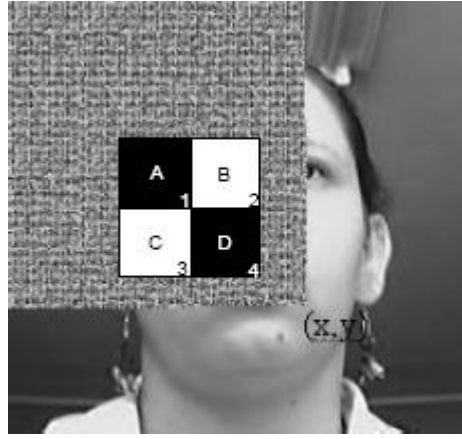


Figura 15: Convolución del filtro Haar con una imagen integral dividida en 4 sub-regiones, donde 1,2,3 y 4 son los resultados de la imagen integral[6]

2.2.2 Asignación de Orientación

Una vez realizada la convolución y seleccionados los puntos máximos, se debe disponer de la orientación y la descripción de la vecindad. Se establece un descriptor para cada punto con la ayuda de los filtros Haar que nos brindan una solución en los ejes x y y en una región circular de $6s$ usado por el autor [10], siendo s la escala del punto de interés.

Con los cálculos realizados para todos los vecinos se estima la orientación dominante, se procede a sumar todos los resultados dentro de una ventana deslizante que cubre un ángulo de $\pi/3$ recomendado en [10], ver la figura 16

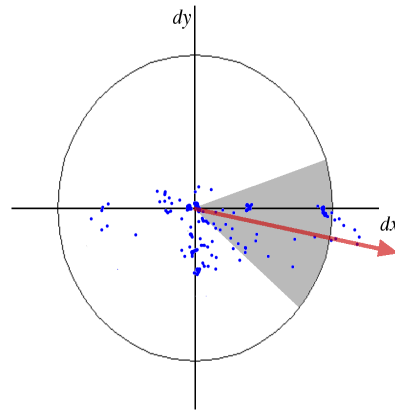


Figura 16: Asignación de orientación [10]

2.2.3 Creación del descriptor

Se crea una región cuadrada centrada en el punto de interés de tamaño $20s$ como se observa en la figura 17. La región es dividida en sub-regiones de 4×4 , en cada una de las cuales se calculan las respuestas de Haar (figura 18) siendo estas d_x y d_y , para cada sub-región se suman los resultados así como sus valores absolutos $|d_x|$, $|d_y|$ como se muestra en la figura 19 quedando formado un vector para cada una de



Figura 17: Imagen con las regiones cuadradas centradas alrededor de los puntos de interés de tamaño $20s$ [10]

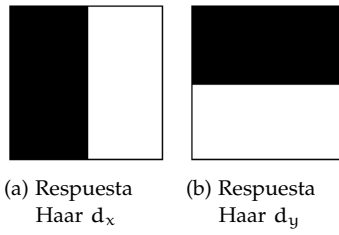


Figura 18: Respuestas Haar de tamaño de $2s$, la parte de color negro representa un peso de -1 y el blanco 1 [10]

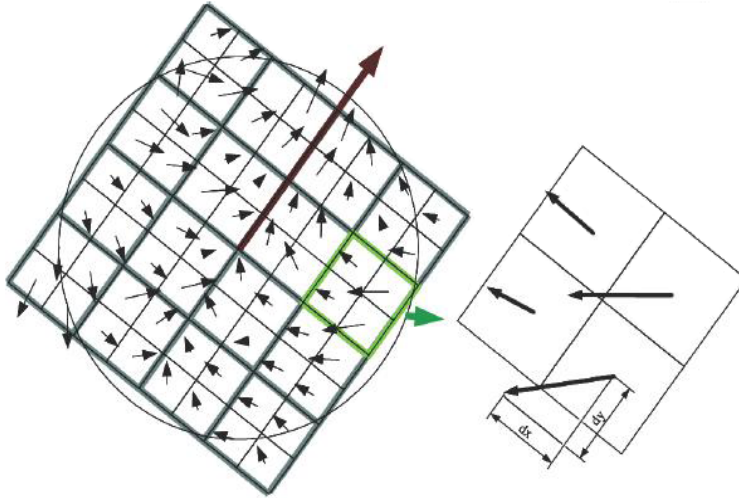


Figura 19: Respuestas de Haar en cada una de las Sub-regiones alrededor del punto de interés

las regiones presentado en la ecuación 2.7, los puntos resultantes son invariantes a la escala.

$$v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|) \quad (2.7)$$

De esta forma se obtiene un descriptor de 64 valores para cada punto identificado.

En la figura 20 se presenta una imagen con los resultados obtenidos después de aplicar el método SURF:



Figura 20: Resultado aplicando el método SURF[28]

2.3 APLICACIONES

2.3.1 Reconocimiento de imágenes móviles

Se usa en teléfonos móviles con el software Kooaba que implementa internamente SURF, y a su vez permite analizar imágenes para ser buscadas en toda la web dando como resultado la dirección URL donde puede ser encontrada la imagen analizada. Esta aplicación es utilizada para sistemas como Iphone y Android ³.

2.3.2 Reconocimiento de rostros

Procesa imágenes de rostros que son almacenados en bases de datos. Como es el caso de un robot que utiliza SURF para procesar imágenes y almacenar altos índices de reconocimiento[13]. El robot por medio de una cámara integrada, captura la región del rostro y verifica si el rostro capturado coincide con alguna imagen que se encuentre almacenada en la base de datos. Lo novedoso es que se puede hacer el reconocimiento de rostro sin tener en cuenta la expresión facial ni los cambios de iluminación, etc.

2.3.3 Estimación de distancia recorrida por medio de robot

Se basa en la obtención de puntos para calcular la posición a partir de la distancia recorrida y la velocidad con la que avanza el robot. Por medio de una cámara el robot captura una serie de imágenes que serán procesadas utilizando SURF para la extracción de puntos de referencia entre las dos imágenes y así calcular la posición y movimiento que debe realizar el robot [7].

³ Fuente: <http://www.xatakamovil.com/aplicaciones/kooaba-planta-cara-a-google-goggles-en-iphone-y-android>, consultado el: 30/01/2012

2.4 VARIANTES SURF

2.4.1 SURF-128

Realiza el mismo proceso que el método SURF, su diferencia radica en la dimensión de los descriptores.

En SURF-128 los valores de d_x , $|d_x|$, d_y y $|d_y|$ son calculados de forma separada [10], doblando de esta manera el vector de dimensión de 64 valores originalmente presentado en SURF a 128.

SURF-128 es el descriptor más distintivo [9], siendo más rápido para cálculos, pero más lento en realizar *matches* debido a su dimensionalidad.

2.4.2 U-SURF (Upright-SURF)

Esta versión de SURF es únicamente invariante a cambios de escala según [24], siendo de gran utilidad para robots que realizan movimientos sobre superficies planas en los que la cámara rota exclusivamente sobre el eje vertical. Es más robusto que SURF pero solo en una rotación de $\pm 15^\circ$.

Al igual que SURF, para la obtención de los descriptores genera un vector de dimensión de 64 valores, en los cuales incluye la escala y rotación pero no la posición. Resulta ser más rápido que SURF ya que no realiza cálculos de orientación, es decir U-SURF omite el proceso de "Asignación de Orientación" [9].

2.5 HERRAMIENTAS DISPONIBLES

2.5.1 KOOABA

Es un buscador visual usado para el reconocimiento de imágenes móviles basta con solo tomar una foto de una imagen y esto nos devolverá los posibles resultados como links a Youtube, Ebay, etc. Este proceso de reconocimiento de imágenes lo realiza utilizando el método SURF ⁴.

2.5.2 Open CV

Es una herramienta para el tratamiento de imágenes que tiene muchos métodos entre los cuales el de interés para nuestro caso de estudio es SURF que se lo puede encontrar dentro del fichero Samples con el nombre `find_obj.cpp`. Basta con compilarlo y ejecutarlo para probar esta poderosa herramienta.

2.5.3 CUDA SURF

El método SURF es utilizado en GPU para tarjetas NVidia que tienen la tecnología CUDA para conseguir un mayor rendimiento o velocidad en cálculos de procesamiento [28], trabaja incluso con imágenes HD.

La tecnología CUDA hace alusión a un compilador así como a un conjunto de herramientas de desarrollo, que permite a los programadores usar una variación del lenguaje de programación C para codificar

GPU: Unidad de Procesamiento Gráfico.

CUDA: Arquitectura de Dispositivos de Computo Unificado.

HD: High Definition (Alta Definición).

⁴ Fuente: <http://www.xatakamovil.com/aplicaciones/kooaba-planta-cara-a-google-goggles-en-iphone-y-android>, consultado el: 30/01/2012

algoritmos en GPU de nVidia, y de esta forma aprovechar su gran potencia e incrementar el rendimiento del sistema ⁵.

⁵ Fuente: <http://es.wikipedia.org/wiki/CUDA>, consultado el: 30/01/2012

Part III

WAVELETS

3.1 INTRODUCCIÓN

A partir del año 1807 Joseph Fourier demuestra que, se puede expresar como una suma infinita de senos y cosenos cualquier función periódica, de aquí se comienza a materializar la idea de las wavelets. Con Dennis Gabor una señal es separada en “paquetes de frecuencia-tiempo”. Jean Morlet fracciona una señal sísmica en una forma de wavelets constantes y con la ayuda de Alex Grossman en el año de 1984 generan un modelo en el cual aparece por primera vez el término de wavelets.

A partir de una “wavelet madre” se pueden dilatar, comprimir o desplazar para construir diferentes familias de wavelets las mismas que no se verán afectadas puesto que siempre va a mantener la misma forma. Muchas de las wavelets creadas son distintas alternativas de la wavelet madre.

Con el estudio realizado por Mallat, ya se consiguió realizar una análisis de la wavelet sin saber la fórmula de la wavelet madre, consiguiendo de esta forma una wavelets mas sencillas, reduciendo así las operaciones de cálculo.

Posteriormente se descubrió otro tipo de wavelet, que esta vez ya no eran solo ortogonales sino que se construían por medio de simples y cortos filtros digitales, resultando fáciles de programar. Es así que hoy en día contamos con métodos mucho más sencillos para análisis de imágenes.

Se puede decir que los wavelets son métodos efectivos para análisis de imágenes ya que son capaces de capturar información espacial con sus características visuales, trabajan de manera especial con señales de pulsos que no suceden de manera periódica. Ofrecen un mejor filtrado de ruido presentando resultados computacionalmente eficientes y rápidos frente a otros filtros como Fourier.

Un dato impresionante de las wavelets es que adoptan las características idénticas que nuestros ojos son capaces de captar, rehacen una imagen trazando bordes del mismo modo que hacemos los humanos cuando realizamos el bosquejo de un dibujo. Algunos investigadores han mencionado que la semejanza entre la visión humana y las transformadas de wavelets no es una simple casualidad, ya que las neuronas propagan señales visuales idénticas a las wavelets [30].

Las transformadas de wavelets que se aplica para funciones continuas vienen dadas por la siguiente ecuación:

$$S(a, \tau) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{a}} \psi^* \left(\frac{t - \tau}{a} \right) \cdot s(t) \cdot dt \quad (3.1)$$

donde τ representa las traslaciones (figura 21b) y a las dilataciones (figuras 21c, 21d y 21e).

Existe un sin número de wavelets, por citar unas cuantas: wavelets de Gabor, Daubechies, Haar, Morlet. Por ejemplo las wavelets de Gabor son de gran utilidad para extraer características de imágenes médicas

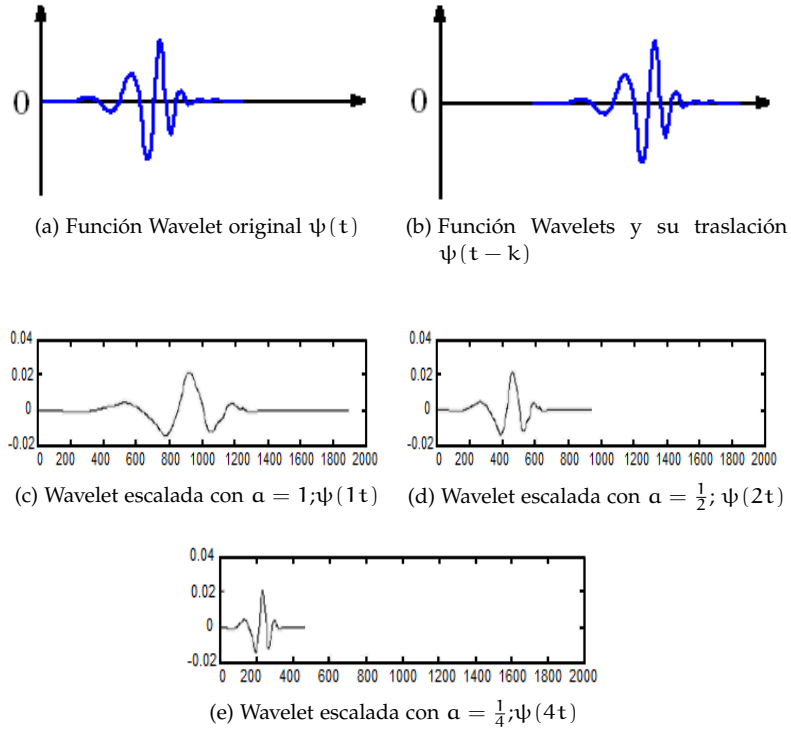


Figura 21: Función Wavelet con sus escalaciones y traslaciones correspondientes, imagen tomada de [11]

basadas en sus contenidos, los mismos que utilizan como característica principal la textura puesto que la imagen exhibe muestras duplicadas dentro de los órganos examinados del paciente. Los filtros de Gabor permiten analizar imágenes y extraer características invariantes a la rotación y traslación.

3.2 FUNDAMENTO TEÓRICO

3.2.1 Transformada Wavelet Continua

La transformada wavelet continua se aplica para el análisis de señales discretas, como imágenes, sonidos, etc

Siendo $g_a(t) = g(t/a)$ una función de escalamiento y $g_b(t) = g(t - b)$ una función de traslación, al aplicar estas dos operaciones en paralelo tenemos la función $g_a^b(t)$ como se muestra en la ecuación 3.2:

$$g_a^b(t) = g\left(\frac{(t - b)}{a}\right) \quad (3.2)$$

si esta cumple con las propiedades elementales especificadas en la ecuación 3.3,

$$\int_{-\infty}^{\infty} g(t)dt = 0 \quad ; \quad \int_{-\infty}^{\infty} g^2(t)dt = 1 \quad (3.3)$$

se podría decir que $g(t) = \psi(t)$, representando $\psi(t)$ la wavelet madre, caso contrario es considerada como una simplemente como una wavelet [32].

$$\psi_a^b(t) = \frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right) \quad (3.4)$$

Dentro de las propiedades de las waveletes se va a listar las siguientes de acuerdo a lo citado en [14]:

- Propiedad de Admisibilidad.

Satisface el criterio de admisibilidad si:

$$C_\psi = \int_{-\infty}^{+\infty} \frac{|\Psi(f)|^2}{|f|} df < \infty \quad (3.5)$$

$\Psi(f)$ es la transformada de Fourier de $\psi(x)$ (wavelet madre).

Al cumplir con este criterio se logra que la función wavelet: 1) esté perfectamente localizada en el tiempo o espacio, 2) se pueda reconstruir de forma eficiente, 3) con su transformada de Fourier tenga un espectro de tipo pasa-banda con rápido decaimiento hacia el infinito (ecuación 3.6).

$$\Psi(0) = 0 \Rightarrow \int_{-\infty}^{+\infty} \psi(x) dx = 0 \quad (3.6)$$

- Regularidad

Esta propiedad indica que la wavelet madre debe ser localmente suavizada y concentrada en dominios de tiempo-frecuencia para lo cual se debe hacer que la transformada wavelet decrezca vertiginosamente. Como consecuencia de esta condición aparece el concepto de momentos de desaparición, siendo calculados de la siguiente manera:

- Si se expande la transformada wavelet en las series de Taylor con $t = 0$ se tiene:

$$T(a, 0) = \frac{1}{\sqrt{a}} \left[\sum_{p=0}^n f^{(p)}(0) \cdot \int \frac{t^p}{p!} \cdot \psi\left(\frac{t}{a}\right) \cdot dt + O(n+1) \right] \quad (3.7)$$

siendo a la escala, $f^{(p)}$ la p -ésima derivada de f , $O(n+1)$ significa el resto de la expansión.

Se define los momentos de wavelet por $M_p = \int t^p \psi(t) dt$ quedando expresada la ecuación de la siguiente manera:

$$T(a, 0) = \frac{1}{\sqrt{a}} \left(f(0)M_0a + \frac{f^{(1)}(0)}{1!}M_1a^2 + \frac{f^{(2)}(0)}{2!}M_2a^3 + \dots + \frac{f^{(n)}(0)}{n!}M_na^{n+1} + O(a^{n+2}) \right) \quad (3.8)$$

mediante la propiedad de admisibilidad al tener $M_0 = 0$, el primer termino de la ecuación 3.8 se vuelve 0. Si se opera de igual manera hasta $M_n = 0$ entonces, los coeficientes de la transformada wavelet tendrá un decaimiento exponencial (α^{n+2}) para una señal suave $f(t)$.

3.2.2 Transformada Wavelet Discreta(DWT)

Al aplicar la DWT a una imagen se obtiene cuatro tipos de coeficientes que son: aproximaciones, detalles horizontales, verticales y diagonales como se puede apreciar en la figura 22

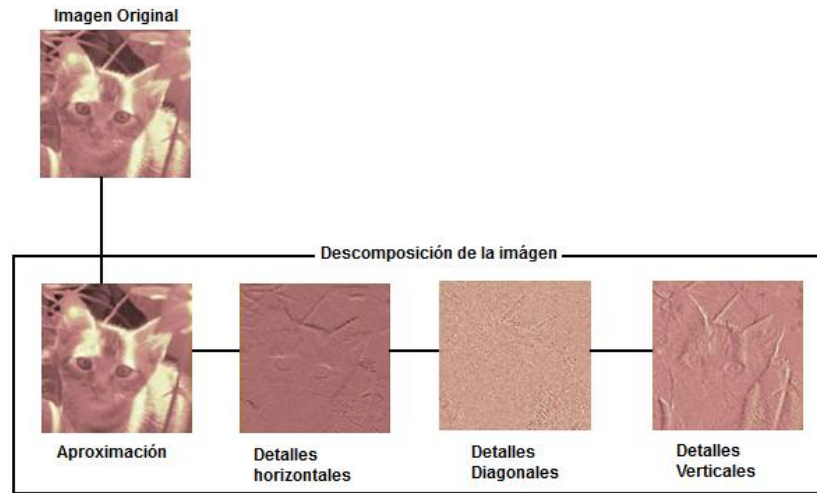


Figura 22: Tipos de coeficientes al aplicar la DWT [26]

La transformada wavelet discreta se aplica para la codificación de señales. Esta proporciona la suficiente información ofreciendo una reducción de tiempo en cómputo. A comparación con la transformada wavelet continua es más fácil su implementación [23].

Para que pueda trabajar el computador, realizando cálculos es necesario discretizar las transformadas por lo tanto al discretizar una variable es reemplazar el muestreo continuo por un conjunto finito de valores tal como se indica en [31].

En la transformada discreta selecciona un conjunto de escalas y posiciones para muestrear las señales y de esta forma discretizar los valores. En sí la transformada wavelet discreta es la transformada wavelet continua discretizada, por lo tanto la wavelet continua puede ser expresada de la siguiente manera:

$$\psi_{j,k}(x) = \frac{1}{\sqrt{s_0^j}} \psi\left(\frac{x - k\tau_0 s_0^j}{s_0^j}\right) \quad (3.9)$$

siendo j, k factores enteros de escalado y traslación, $s_0 > 1$ un paso predeterminado de dilatación, τ_0 la traslación que depende de la dilatación.

Con la ecuación 3.9 se logró muestrear el espacio tiempo-escala a intervalos discretos. Al dar valores a $s_0 = 2$ y $\tau_0 = 1$ se está tratando de un muestreo diádico (ver la figura 23) estos valores son naturales

Diádico: Según el diccionario digital la palabra diádico significa "Pareja de dos seres o cosas estrecha y especialmente vinculados entre sí"

para los sistemas computacionales, así como para el oído humano y la música [17].

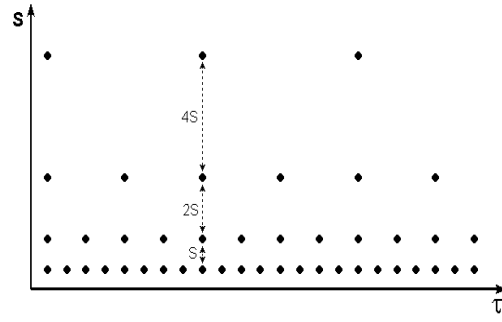


Figura 23: Localización de las wavelets discretas en tiempo-escala con muestreo diádico, tomada de [23]

Al utilizar las wavelets discretas para transformar una señal continua da como consecuencia una serie de coeficientes tomando el nombre *descomposición de series de wavelets*. Así como se descompone la señal es necesario la reconstrucción. Ingrid Daubechies [17] demostró que la condición para la reconstrucción estable los coeficientes de las wavelets deben estar dentro de dos límites positivos que se muestran en la ecuación 3.10:

$$A\|f\|^2 \leq \sum_{j,k} |\langle f, \psi_{j,k} \rangle|^2 \leq B\|f\|^2 \quad (3.10)$$

donde $\|f\|^2$ es la energía de $f(t)$, A y B son independientes de $f(t)$, siendo $A > 0$ y $B < \infty$.

Cuando la ecuación 3.10 se satisface se denomina trama con límites A y B . Si A y B son iguales, la trama es cerrada y las wavelets discretas proceden como funciones ortonormales. Si A y B son diferentes la reconstrucción aún es posible pero para realizarlo se deberá utilizar una trama dual.

Por último, para reducir toda la redundancia de la transformada wavelet se debe usar wavelets discretas ortonormales. Se denomina a una transformada wavelet discreta redundante si una señal de N muestras es transformada en $M > N$ muestras, por lo contrario se llama no redundante cuando la base wavelet es ortogonal.

Las wavelets discretas pueden hacerse ortogonales desde su propias traslaciones y escalas si cumple con la ecuación 3.11:

$$\int \psi_{j,k}(t) \psi_{m,n}(t) dt = \begin{cases} 1 & \text{si } j = m \text{ y } k = n \\ 0 & \text{en otro caso} \end{cases} \quad (3.11)$$

Una señal puede reconstruirse por medio de la suma de las funciones wavelets ortogonales base utilizando la ecuación 3.12, esta es la transformada wavelet inversa:

$$f(t) = \sum_{j,k} \gamma(j,k) \psi_{j,k}(t) \quad (3.12)$$

La ortogonalidad no siempre puede ser utilizada para todas las wavelets ya que en algunos casos la redundancia ayuda a reducir el ruido o mejorar la invariancia al cambio de la transformación.

Filtro pasa-banda

A pesar de haber eliminado la redundancia, no se puede utilizar las wavelets ya que se presentan dos inconvenientes que son:

- Reducir el número de wavelets en la transformada wavelet para guardar en soluciones analíticas.
- Al tener la wavelet discreta todavía se tiene un número infinito de escalamientos y traslaciones para el cálculo de la transformada wavelet.

Las traslaciones de las wavelets son limitadas por la duración de la señal. En la ecuación 3.6 la wavelet tiene un espectro como pasa-banda. En la teoría de Fourier, la compresión en el tiempo es igual a la expansión del espectro y desplazado hacia arriba en frecuencia (ecuación 3.13)

$$F\{f(at)\} = \frac{1}{|a|} F\left(\frac{\omega}{a}\right) \quad (3.13)$$

La expansión en tiempo de la wavelet en un coeficiente de 2 comprimirá el espectro de frecuencia en un factor 2, así como desplazará todos los elementos de frecuencia hacia abajo en un factor tal como se puede apreciar en la figura 24

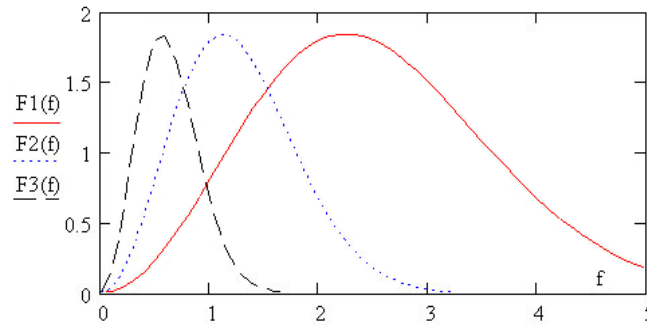


Figura 24: Expansión y desplazamiento en factor 2, imagen tomada de [23]

Usando esta característica se puede cubrir el espectro finito de la señal con el espectro de wavelets escaladas de la misma manera que se cubre la señal en el dominio del tiempo con wavelets trasladados como menciona el autor en [23].

El espectro de la wavelet estirado debería enlazarse el uno con el otro para de esta forma tener un buen espectro de la señal tal como se muestra en la figura 25.

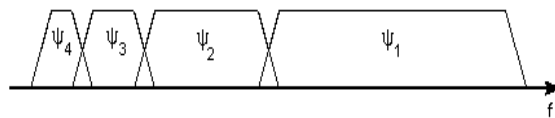


Figura 25: Espectros enlazados secuencialmente. [23]

Se podría decir que si una wavelet se ve como un filtro pasa-banda, por lo tanto una cadena de wavelet dilatados se vería como un banco de filtros pasa-banda.

Existe un requerimiento el cual es que la señal debe tener energía finita, al tener energía infinita no se puede cumplir todo el espectro de frecuencia y su duración temporal. Para ello se utiliza la siguiente fórmula:

$$\int_{-\infty}^{\infty} |x(t)|^2 dt < \infty$$

Es importante mencionar que las señales naturales tienen energía finita.

Función de escalado

Al tener la señal con energía el siguiente paso es sumar hasta el infinito los productos entre los coeficientes de la DWT, para así obtener la señal original de forma exacta, esto se logra mediante el espectro paso-bajo o función de escalado.

La función de escalado puede ser expresada como una señal con espectro paso-bajo, pudiendo descomponerse en las componentes wavelets y ser expresada como la ecuación 3.14:

$$\phi(t) = \sum_{j,k} \gamma(j,k) \psi_{j,k}(t) \quad (3.14)$$

Existe un gran inconveniente cuando se estira la wavelet solo se cubre la mitad del espectro por lo que se necesita un número infinito de wavelets, la solución es no intentar cubrir el espectro total, por lo cual se va limitar utilizando un espectro paso-bajo tal como se muestra en la figura 26.

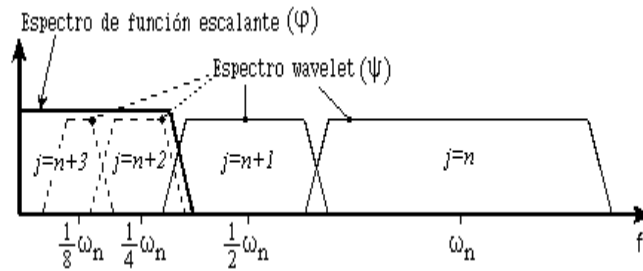


Figura 26: Función escalado, imagen tomada de [23]

Cuando se utiliza una función de escalado se pierde información, el espectro paso-bajo posee la condición de admisibilidad presentada en la ecuación 3.15:

$$\int_{-\infty}^{\infty} \phi(t) dt = 1 \quad (3.15)$$

Esta condición demuestra que en el momento 0-ésimo de la función no se anula. Se puede deducir que "si una wavelet puede ser visto como

un filtro pasa-banda y la función de escalado como un filtro paso-bajo, entonces serie de wavelets trasladados juntos con la función de escalado pueden ser vistos como un banco de filtros” [23].

Codificación sub-banda

Al poseer una señal de banda limitada cumpliendo con la ecuación 3.16 se puede muestrear las señal de manera uniforme para que quede de la forma presentada en la ecuación 3.17:

$$F\{u(x)\} = U(f) = 0 \quad \text{para} \quad |f| \geq f_{\max} \quad (3.16)$$

$$u(i\Delta x) \quad i = 0, 1, \dots, N-1 \quad f_{\max} \leq f_N = \frac{1}{2\Delta x} \quad (3.17)$$

siendo f_N la frecuencia de Nyquist.

3.3 APLICACIONES

3.3.1 *Análisis de señales*

Procesamiento de señales de acústica: las señales acústicas se consideran como la superposición de sinusoides modulados por exponenciales decrecientes [32], en estas señales se comprueba si existe o no la presencia de ruido aditivo y emisiones de energía elástica. En definitiva se debe estimar sus localizaciones, frecuencias, radio, fases, y describir su estructura de superposición. Es por tal motivo que se utiliza las wavelets de Gabor para procesar este tipo de señales.

3.3.2 *Aplicación al diseño textil*

En la producción textil existe un gran volumen de información visual que tiene necesidades de realizar búsquedas, exploración y recuperación de la información, se ha fomentado la idea de tratar los elementos básicos como la textura, color y forma mediante CBIR que utiliza wavelets como componente para implementar análisis multiresolución y así extraer características o realizar indización de las imágenes. Se realiza búsquedas de las imágenes con un criterio de similitud, es decir no tiene que dar un resultado de coincidencia exacta entre dos imágenes [20].

CBIR: Content-based
image retrieval (Recu-
peración de imágenes
basada en contenidos)

3.3.3 *Reconocimiento de rostros*

En el reconocimiento de rostros mediante wavelets de Gabor realiza un análisis de la señal que se va representando en cada momento de tiempo. Según [2] los wavelets de Gabor son el producto de una senoide con una función gaussiana, siendo el objetivo limitar los valores de cada pixel, aproximados a un punto característico del rostro que contribuye a la convolución. Los coeficientes de la convolución de la orientación y escala sobre cada uno de los puntos característicos forman un vector de características locales el cual contiene la ubicación del contorno del rostro, ojos, nariz, boca y cejas.

3.3.4 Extracción de características en la identificación de Patologías de ECG

El análisis de las señales ECG radica en la exploración de parámetros como duración, regularidad y amplitud que proporcionan información del estado funcional y orgánico del paciente en el diagnóstico clínico. Mediante las wavelets se extraen características orientadas a diversos tipos de irregularidades cardíacas, representa una mayor confiabilidad en la detección de QRS calculando características como la duración y amplitud de cada latido con menos probabilidad de que un ruido de gran amplitud y corta duración sea confundido con algún segmento de QRS. Con el uso de la transformada wavelet se obtiene características en conjunto de los coeficientes de aproximación de la señal, por ello la transformada discreta wavelet obtiene dos conjuntos de coeficientes, los de aproximación y los de detalle, a través de la aplicación de la convolución de dichos coeficientes con un filtro paso bajo y paso alto.

La transformada wavelet [21] presenta ventajas como la reducción de ruido, invariabilidad en la frecuencia de muestreo, extracción de parámetros mediante los coeficientes, clasificación, comprensión y transmisión de señales.

ECG: Electrocardiograma

QRS: Marca característica de la señal electrocardiográfica

3.4 VARIANTES WAVELETS

3.4.1 Waveletes Haar

Son las primeras wavelets, fueron propuestas por Alfred Haar en el año de 1909, de allí su nombre, son conocidas también como la función esca-la según se señala en [29], en la figura 27 se muestra su representación descrita en la ecuación 3.18.

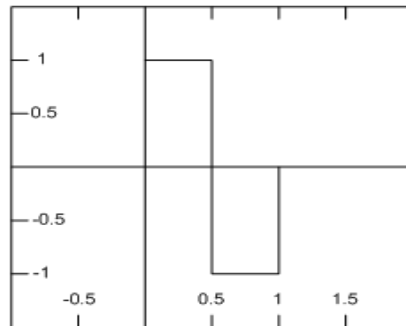


Figura 27: Función Haar

$$\psi(x) = \begin{cases} 1 & 0 \leq x < \frac{1}{2} \\ -1 & \frac{1}{2} \leq x < 1 \\ 0 & \text{en cualquier otro caso} \end{cases} \quad (3.18)$$

Esta wavelet por ser sencilla y antigua es la más utilizada para el análisis de señales usando transformadas discretas y continuas.

3.4.2 Daubechies

Biortogonalidad: wavelets simétricas ó funciones par o impar

Orden: denota el número de coeficientes del filtro que tienen las wavelets de Daubechies

Las wavelets de Daubechies poseen características de ortogonalidad y biortogonalidad así como también se logra obtener las transformadas wavelets discretas (DTW) y las continuas, fueron propuestas por Ingrid Daubechies en la década de los 80's [29]. Cada wavelet tiene un orden N que indica su suavidad, siendo N un numero entero positivo representando el numero de filtros. Existen wavelets Daubechies de distinto orden, por ejemplo db4 representa a la orden 4 tal como se muestra en la figura 28.

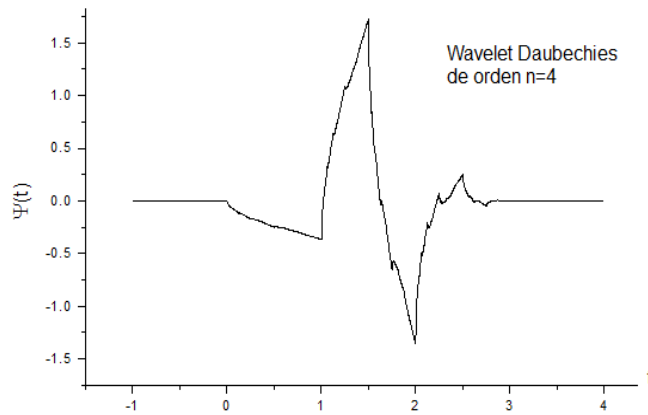


Figura 28: Wavelet Daubechies de orden 4 tomada de [11]

3.4.3 Gabor

Se fundamentan en un conjunto de filtros lineales e invariantes que permiten ser captados en determinada orientación y frecuencia, se caracterizan por su respuesta al impulso basado en un núcleo gaussiano modulado por una senoide como se muestra en la ecuación 3.19.

$$W(x, y, \theta, f, \phi, \sigma, \gamma) = e^{-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}} \cos(2\pi f x' + \phi) \quad (3.19)$$

con $x' = x \cos(\theta) + y \sin(\theta)$ y $y' = -x \sin(\theta) + y \cos(\theta)$, donde f representa la frecuencia central de la senoide, θ la orientación, ϕ el desfase en grados, ϕ la desviación estandar, γ es definida para mantener la frecuencia y suavidad constante.

En la figura 29 [18] se puede ver como trabajan las wavelets de Gabor en el reconocimiento de rostros, donde obtiene la parte real 29b de la imagen y la magnitud 29c

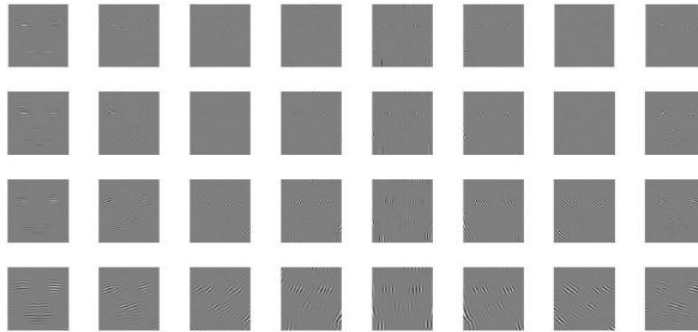
3.5 HERRAMIENTAS DISPONIBLES

3.5.1 Scilab wavelet toolbox

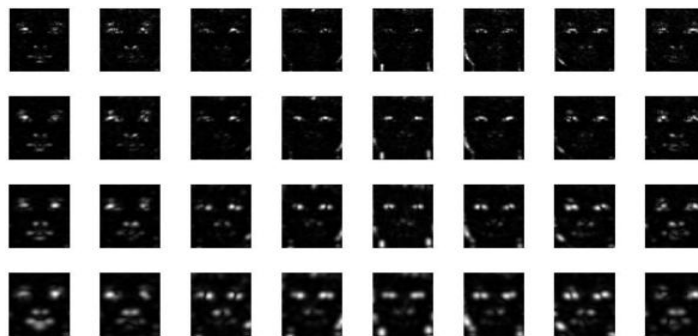
Es un paquete de software libre utilizado para el procesamiento de señales, esta desarrollado en C. Scilab está desarrollado para la educación en general y también es posible usarlo en la investigación en el procesa-



(a) Imagen original del rostro



(b) Parte real de la imagen



(c) Magnitud de la imagen

Figura 29: Wavelets Gabor en el reconocimiento de rostros

miento de señales, es decir, realiza el análisis de señales ultrasonido, señales sísmicas, imágenes para diagnóstico médico, etc.¹

Scilab permite realizar lo siguiente:

- Transformada rápida wavelet discreta
- Transformada wavelet estacionaria
- Transformada wavelet continua
- Wavelet Daubechies
- Descomposición y reconstrucción de señales

3.5.2 Wavelab

Wavelab es una biblioteca de Matlab, que posee una gran variedad de funciones, es utilizado para el análisis de wavelets, senos, cosenos, etc.

¹ Fuente: <http://scwt.sourceforge.net/>, consultado el: 23/01/2012

Consta de mas de 1200 archivos que están documentados, se basan en secuencias de comandos². Wavelab es utilizada para la enseñanza en cursos de análisis wavelets, es la base para la investigación de wavelets utilizada por muchos autores.

Wavelab tiene las siguientes técnicas:

- Transformada wavelet ortogonal, biortogonal
- Traslación wavelet invariante
- Interpolación de la transformada wavelet

3.5.3 *LabView Add-On: Procesamiento avanzado de señales*

LabView es una herramienta gráfica de programación creada por National Instruments³, permite realizar pruebas de control y diseño, es comúnmente usado por ingenieros electrónicos y científicos, mediante Labview se puede realizar las siguientes tareas:

- Diseño de controladores como simulación y prototipaje rápido.
- Robótica
- Visión artificial y control de movimiento
- Domótica y redes de sensores inalámbricos

Mediante LabView Add-On permite escoger una transformada wavelet adecuada o diseñar una wavelet a través de una interfaz gráfica de usuario para así descubrir la mejor wavelet para la aplicación que se desea implementar ⁴.

² Fuente: http://www-stat.stanford.edu/~wavelab/Wavelab_850/intro.html, consultado el 23/01/2012

³ Fuente: <http://es.wikipedia.org/wiki/LabVIEW>, consultado el: 23/01/2012

⁴ Fuente: <http://www.ni.com/analysis/wavelet.htm>, consultada el: 23/01/2012

Part IV

SELECCIÓN DE LA HERRAMIENTA DE EXPERIMENTACIÓN

SELECCIÓN DE LA HERRAMIENTA DE EXPERIMENTACIÓN

En los capítulos anteriores se estudiaron tres métodos que son SIFT, SURF y Wavelets, y se investigó su funcionamiento. Existen muchas herramientas disponibles que utilizan los métodos antes mencionados, sin embargo nos enfocaremos en una herramienta basada en el método SURF como ya se explicó detalladamente en el capítulo 2, pues este método de recuperación de imágenes basado en contenidos es más robusto y el objetivo es comprobar la efectividad del mismo.

4.1 REVISIÓN DE LAS HERRAMIENTAS DISPONIBLES

Existen muchas herramientas disponibles que utilizan el método SURF, algunas de código libre y otras que son comerciales, a continuación se mencionarán las siguientes herramientas:

4.1.1 *Método Original*

Este método es la versión original de SURF y la raíz de las implementaciones posteriores, lo podemos encontrar en la siguiente dirección:

<http://www.vision.ee.ethz.ch/~surf/SURF-V1.0.9-WinDLLVC8.tar.gz>

Este código lo podemos utilizar libremente y modificar los parámetros que nosotros creamos convenientes, adecuándolo según nuestros criterios para la obtención de los puntos relevantes de una imagen.

4.1.2 *OpenSurf*

Al igual que el código original este es utilizado para detectar puntos de interés entre imágenes. El autor, Chris Evans presenta en su página “Visión por Computador y Procesamiento de Imágenes”, el código del método SURF desarrollado en C++ , C# y el repositorio OpenSurf que lo podemos encontrar en la siguiente dirección:

<http://www.chrisevansdev.com/computer-vision-opensurf.html>

4.1.3 *OpenSurfCL*

Es una implementación paralela de OpenSurf con OpenCL en GPU (Unidad Gráfica de Procesamiento) el mismo que lo podemos encontrar en la siguiente URL:

<http://sourceforge.net/projects/opensurfcl/>

4.1.4 *OpenCV SURF*

Dentro de OpenCV se cuentan con librerías que permiten implementar el código en C++, así mismo es posible encontrarlo en la siguiente dirección:

http://opencv.willowgarage.com/documentation/cpp/feature_detection.html#surf

4.1.5 *SurfMex*

Esta es una interfaz Matlab de SURF para la plataforma de Windows, se compila la clase startup.m el mismo que llama a OpenCV para poder ejecutar el programa. El código se lo puede encontrar en la siguiente dirección:

<http://www.maths.lth.se/matematiklth/personal/petter/surfmex.php>

4.1.6 *OpenSURF Matlab*

Esta implementación está desarrollada en Matlab bajo licencia BSD, la cual no es solo una interfaz de conexión OpenCV, como en el caso anterior. Este código en sí es toda la aplicación de SURF basada en la implementación realizada por Chris Evans. Es posible descargar el código de la siguiente dirección:

<http://www.mathworks.com/matlabcentral/fileexchange/28300>

4.1.7 *Libmv SURF*

Es una implementación para la extracción de puntos y *matching* desarrollado en el lenguaje de C++, algunas de las librerías las podemos encontrar en la siguiente dirección:

<http://code.google.com/p/libmv/>

4.1.8 *JOpenSURF*

Es una implementación de SURF realizada en el lenguaje de programación Java. Se puede encontrar el código en la siguiente dirección:

<http://code.google.com/p/jopensurf/downloads/detail?name=JOpenSurf-2010-11-20.tar.gz>

4.1.9 *Parallel SURF*

Esta implementación de SURF Paralelo utiliza multi-hilos a fin de aligerar el cálculo en máquinas multicore. Existen muchas versiones de este código, pero la mas actual la podemos encontrar en la siguiente dirección:

<http://sourceforge.net/projects/parallelsurf/files/Parallel%20SURF%200.96/>

4.1.10 *FPGA SURF*

Esta herramienta está basada en SURF, es utilizada en los robots autónomos en su primera etapa como reconocimiento de objetos y seguimiento.

En esta aplicación se han realizado cambios en el código para aumentar su capacidad en las FPGA. Se puede descargar de la siguiente dirección:

<http://labe.felk.cvut.cz/~tkrajnik/fpga-surf/restricted.html>

4.2 COMPARATIVA DE LAS HERRAMIENTAS

De las herramientas SURF anteriormente mencionadas se seleccionará solamente tres para el análisis de la tabla comparativa entre SIFT y Wavelets, estas herramientas están desarrolladas en distintos lenguajes de programación como Java, Matlab, C++. Se procederá a realizar una comparativa con los parámetros que se presentan en la tabla 1.

PNG: Portable Network Graphics (Gráficos Portables de Red)

JPEG: Joint Photographic Experts Group (Grupo de Expertos Fotográficos)

GIF: Graphics Interchange Format (Formato de intercambio de gráficos)

TIFF: Tagged Image File Format (Archivos de formato para imágenes etiquetadas)

BMP: Bit Map (Mapa de bits)

PPM: Portable Pix-Map (Mapa de pixeles portable)

SURF			SIFT			Wavelets			
HERRAMIENTAS	JOPENSURF	OPENSURF MATLAB	OPENCV	AUTOPANO-SIFT	MATLAB/C	SIFT++	WAVELAB	WAVEKIT	WALLI
Lenguaje	Java	Matlab	C++	C#	Matlab	C++	Matlab	Matlab	C++
Acceso al código	Si	Si	Si	Si	Si	Si	Si	No	Si
Plataformas que soporta	Microsoft Windows, GNU/Linux, MacOS	Microsoft Windows	Microsoft Windows, GNU/Linux, MacOS	Microsoft Windows, GNU/Linux, MacOS	Microsoft Windows, GNU/Linux, MacOS	Microsoft Windows, GNU/Linux, MacOS	Microsoft Windows, GNU/Linux, MacOS	Microsoft Windows, GNU/Linux, MacOS	Microsoft Windows, GNU/Linux, MacOS
Algoritmos de soporte para el procesamiento de imágenes	No	No	Soporta alrededor de 500 algoritmos	No	No	No	Si	No	Si
Aplicación directa a sistemas CBIR	Si	Si	Si	Si	Si	Si	No	No	No
Tamaño de descriptores	64	64	64	128	128	128	Variante	Variante	Variante
Formato de imágenes que soporta	PNG, JPEG, GIF, TIFF, BMP	JPEG, TIFF, BMP, GIF	Soporta todo tipo de imágenes JPEG, PNG, GIF, TIFF, PPM	JPEG, GIF, PNG, PPM, TIFF, BMP	JPEG, GIF, TIFF, BMP	JPEG, PNG, GIF, TIFF	JPEG, TIFF, BMP, GIF	JPEG, GIF, TIFF, BMP	JPEG, PNG, GIF, TIFF, PPM

Cuadro 1: Tabla comparativa de las herramientas SURF.

4.3 SELECCIÓN Y PRUEBAS DE LA HERRAMIENTA

La herramienta que se utiliza es OpenCV la misma que fue desarrollada por Intel bajo la licencia BSD en el año 1999 y posee una amplia gama de algoritmos y utilitarios para el tratamiento de imágenes.

En la tabla comparativa¹ se demuestra que OpenCV es la mejor herramienta, en virtud de las siguientes consideraciones:

- Provee algoritmos implementados para realizar transformaciones afines sobre una imagen como traslación, rotación, escalado, siendo estas necesarias para llevar a cabo nuestro experimento.
- La implementación de OpenCV se rige a las propuesta original de SURF.
- Proporciona funciones que son utilizadas para el tratamiento de imágenes como la binarización.
- Permite realizar operaciones como invertir matrices ó la multiplicación entre ellas.
- Se puede realizar funciones como lectura y escritura de imágenes.

En OpenCV viene incorporado el programa de SURF con el nombre de `find_obj.cpp`, en la cual se acoplarán los parámetros necesarios para cumplir con el objetivo planteado de medir la precisión de dicha herramienta.

Por el contrario en Java viene desarrollado el código de SURF pero para acoplarlo se necesita otras librerías, así como también en Matlab viene el código de SURF pero por ser software propietario se dificulta su implementación ya que excluye ciertas clases necesarias para su compilación que no son de código abierto.

Para realizar las pruebas de la herramienta OpenCV se va a utilizar dos imágenes, la una viene por defecto dentro del código SURF y la otra es una imagen en la cuál se insertó ruido, en las figuras (30), (31) se presenta el resultado de las pruebas realizadas con SURF.

Como conclusión, en las pruebas realizadas se puede observar que en las figuras 31, 30, la técnica SURF implementada con OpenCV presenta un buen rendimiento y es capaz de detectar correctamente los puntos característicos de las imágenes originales y en las que se ha insertado ruido.



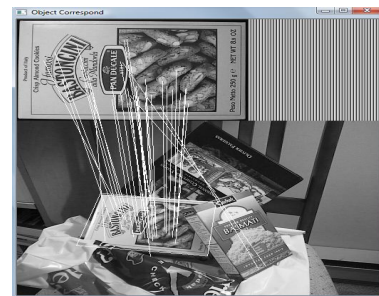
(a) Imagen Original



(b) Imagen con ruido



(c) Detección de puntos

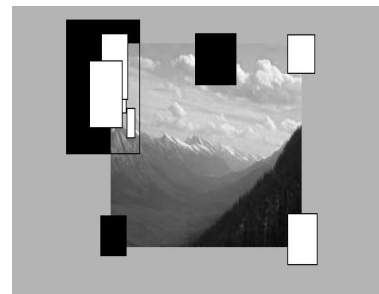


(d) Resultado de comparación de puntos entre las dos imágenes

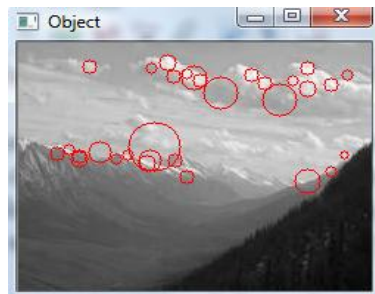
Figura 30: Pruebas de la herramienta con la imagen por defecto



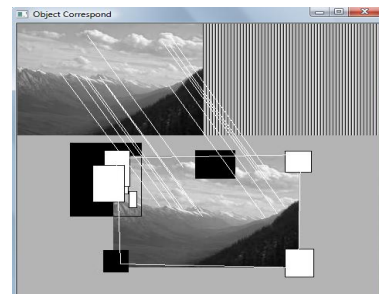
(a) Imagen Original



(b) Imagen con ruido



(c) Detección de puntos



(d) Resultado de comparación de puntos entre las dos imágenes

Figura 31: Pruebas de la herramienta con la segunda imagen

Part V

EXPERIMENTACIÓN

EXPERIMENTACIÓN

5.1 SELECCIÓN DEL CORPUS

En la experimentación se va a trabajar con imágenes de diversa clasificación para comprobar la efectividad del método SURF, por lo tanto estas imágenes son tomadas del Corel DataSet el cual almacena mil imágenes clasificadas en carpetas desde la 0 hasta la 9, cada carpeta contiene 100 imágenes que pertenecen a una sola categoría siendo estas: dinosaurios, rosas, personajes, paisajes, etc.

El Corel DataSet almacena imágenes en formato PPM a color y utilizan 24 bits por pixel, siendo utilizadas para realizar practicas de procesamiento digital puesto que es un formato sin compresión¹. Este tipo de imágenes de formato PPM usa un byte (P6) o un número (P3) para el valor de cada componente (RGB) de cada pixel [35], en la figura 32 se puede observar el encabezado que lleva cada imagen, en nuestro caso las imágenes almacenadas usan PPM binario.

RGB: Red Green Blue
(Imágenes a color rojo, verde, azul)

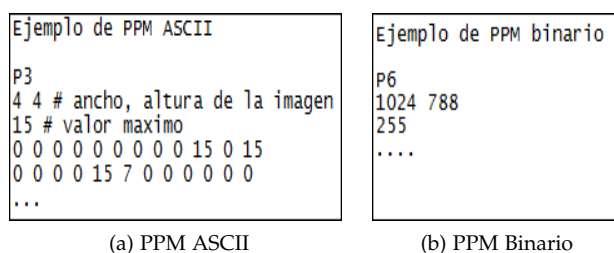


Figura 32: Ejemplos de PPM binario y ASCII

Las imágenes de formato PPM se caracterizan por ser archivos de texto plano pudiendo ser modificadas mediante un procesador de texto.

Al ser imágenes de 24 bits a color RGB, estas varían teniendo las siguientes características según se indica en [15]:

- Grado de iconicidad.- Calidad de identidad de la representación en relación al objeto referido.
- Complejidad.- Según el número de elementos es su simplicidad o dificultad de distinguir al momento de interpretar una imagen.
- Color.- Pueden ser a blanco y negro o a color.
- Calidad técnica.- el contraste, nitidez, saturación.
- Formato y tamaño de la imagen.- Según el formato, el tamaño de la imagen varía pudiendo ser JPG, MPG, BMP, PNG, PPM.

En la figura 33 se presentan algunas imágenes que posee Corel DataSet:

Estas imágenes son procesadas mediante un programa que realiza la traslación, rotación y escalado de cada una, tomando un nombre

¹ Fuente: http://www.lcc.uma.es/~vicente/docencia/docencia/mp_pract_imagen.pdf, consultado el: 12/12/2011

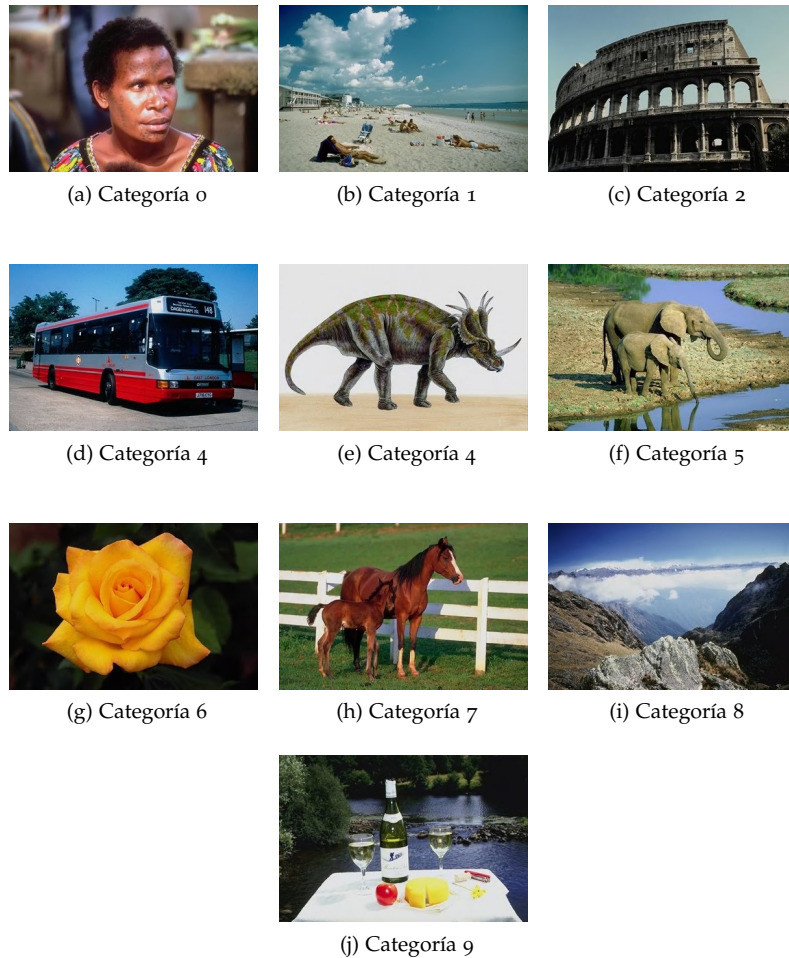


Figura 33: Imágenes del Corel Data Set

diferente según la operación realizada, para posteriormente proceder a la comprobación.

5.2 DISEÑO DEL PLAN DE EXPERIMENTACIÓN

En esta sección se explicará como se ha llevado a cabo el plan de experimentación. En la figura 34, se puede observar cada una de las etapas realizadas. A continuación detallamos el proceso:

- Se toman las imágenes originales para aplicar operaciones de traslación (T), rotación (R), escala (S) y las combinaciones de las mismas, siendo 5 operaciones por cada imagen.
- A continuación las imágenes, tanto originales como transformadas (TRS), son sometidas a un proceso de extracción de características y conteo de puntos de coincidencia. Esta tarea es realizada por el script de extracción de descriptores y comparación, que al recibir dos imágenes, evalúa cuantos puntos de coincidencia tienen las mismas y devuelve este número como resultado.
- Una vez obtenidos los puntos de coincidencia para cada par de imágenes (original-TRS), se procede a realizar un proceso de ordenamiento de mayor a menor. Para efectuar esta tarea se

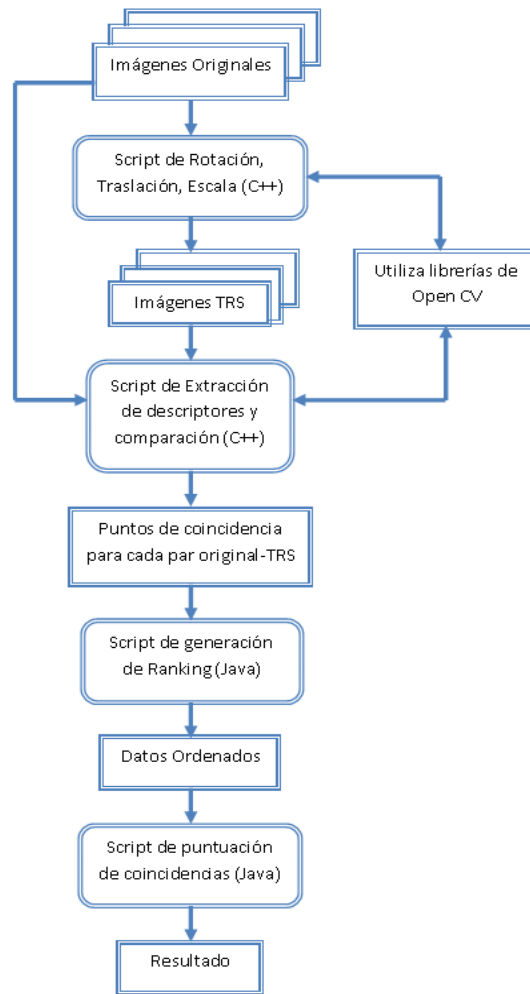


Figura 34: Diagrama del plan de experimentación (los rectángulos ovalados representan procesos y los rectángulos comunes representan datos)

emplea un árbol binario, que dentro de su estructura almacenará los nombres de las imágenes y la cantidad de puntos coincidentes que existen entre ellas.

- Con los resultados obtenidos se procede a establecer en qué comparaciones han existido aciertos. Para ello el script de puntuación de coincidencias verifica que las dos imágenes (original y TRS) pertenezcan a la misma categoría y les asigna un valor de 1. En el caso contrario se asignará 0.

5.3 EJECUCIÓN Y RESULTADOS

A continuación se toma como base el diagrama de la figura 34 para explicar la ejecución del experimento:

1. Tomando en cuenta que el Corel DataSet consta de 1000 imágenes se decidió realizar el experimento solamente con 200, siendo tomadas las dos primeras carpetas por cuestiones de facilidad al momento de realizar la lectura de los archivos.
2. Las 200 imágenes originales son procesadas por el “script de Traslación, Rotación, Escalado” desarrollado en C++ utilizando

librerías de OpenCV, este método realiza las operaciones de traslación (T), rotación (R), escala (S) y las combinaciones de las mismas. Estas operaciones son realizadas con valores aleatorios para cada imagen, para ejecutar el script de TraslacionRotacionEscalado.exe se desarrolló código en Java el mismo que invoca a dicho script así como también va leyendo cada una de las imágenes y aplicando las operaciones correspondientes, dando como resultado 1000 imágenes TRS, es decir se aplican 5 operaciones por imagen siendo estas:

- Traslación, rotación, escalación (T_R_S).
- Traslación, rotación (T_R)
- Traslación, escalación (T_S)
- Rotación, escalación (R_S)
- Rotación (R)

Como se puede ver en la figura 35 se muestra un claro ejemplo de

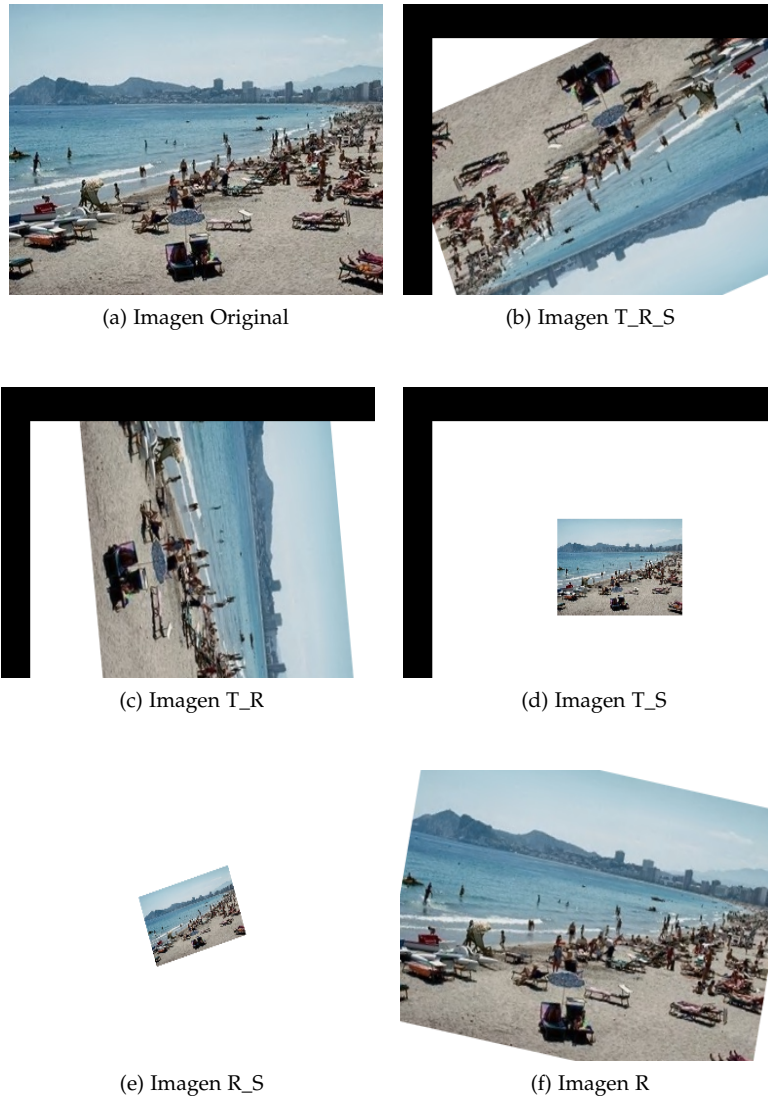


Figura 35: Operaciones aplicadas a la Imagen

las operaciones aplicadas. Cabe aclarar que al realizar la operación

de traslación, OpenCV introduce ruido a la imagen, tal como se puede ver en la figuras 35b, 35c, 35d.

Así mismo, es importante anotar que estas imágenes son guardadas con un nombre específico que nos indica la operación que fue realizada a la imagen.

- T_R_S_o.ppm (imagen trasladada, rotada, escalada)
- T_R_o.ppm (imagen trasladada, rotada)
- T_S_o.ppm (imagen trasladada, escalada)
- R_S_o.ppm (imagen rotada, escalada)
- R_o.ppm (imagen rotada)

3. Mediante el “script de extracción de descriptores y comparación” desarrollado en C++ utilizando librerías de OpenCV, que recibe como parámetros dos imágenes siendo estas la original (36a) y la TRS (36b), pasan por un proceso de binarización y se realiza comparaciones entre las dos imágenes, obteniendo como resultado los puntos de coincidencia como se muestra en la figura 36c. Estos puntos son almacenados en un archivo (análisis.txt) el mismo que consta de: imagen original (36a), imagen TRS (36b), puntos de coincidencia (36c), dando como resultado 200.000 comparaciones, ver ecuación 5.1. El tiempo que tarda en realizar este proceso es de 18 horas, por tal motivo se decidió realizar solamente con las 200 imágenes originales. Al procesar con las 1000 imágenes originales se estima un tiempo de duración de 456 horas ya que daría como resultado 5’000.000 comparaciones, ver tabla 2.

Binarización: pasar una imagen color RGB a escala de grises para un mejor procesamiento

IMÁGENES ORIGINALES	IMÁGENES TRS APLICANDO 5 OPERACIONES	TOTAL DE COMPARACIONES	TIEMPO DE EJECUCIÓN
200	1000	200.000	18 horas
1000	5000	5’000.000	456 horas

Cuadro 2: Tiempo de duración en el proceso de obtención de puntos descriptores

$$\#comparaciones = (\#imágenesOriginales \times \#operaciones) \times \#imágenesOriginales \quad (5.1)$$

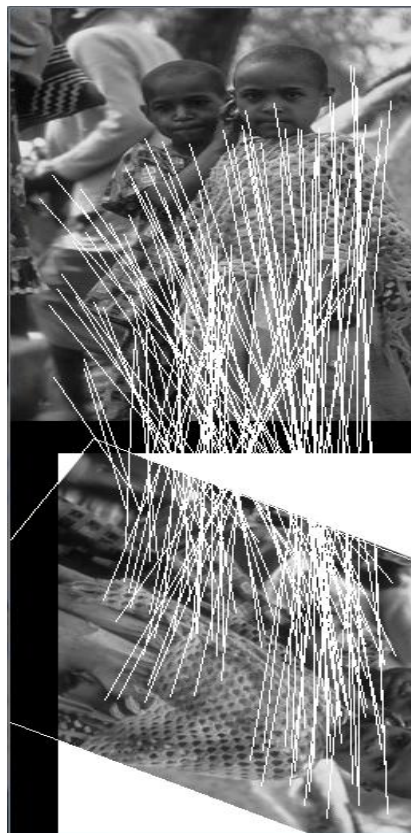
$$\#comparaciones = (200 \times 5) \times 200 = 200,000 \quad (5.2)$$

4. De los resultados obtenidos, se seleccionan las 5 imágenes que tengan el mayor número de puntos de coincidencia respecto a la imagen original, este proceso se realiza mediante el “script de generación de ranking” implementado en Java, el cuál incorpora el código de arboles binarios. El resultado obtenido es almacenado dentro de un archivo (finales.txt) que cuenta con la misma estructura del archivo análisis.txt.



(a) Imagen Original

(b) Imagen trasladada y rotada



(c) Puntos de correspondencia

Figura 36: Detección de puntos de correspondencia aplicando SURF

5. Una vez obtenidos los 5 valores más altos por imagen, se procede a asignar puntuaciones a cada par de imágenes (original-TRS) en las que las comparaciones hayan acertado, para lo cual el “script de puntuación de coincidencias” implementado en Java, verifica si las dos imágenes (original-TRS) corresponden a la misma categoría, entonces asigna un valor de 1, caso contrario se

asigna un valor de 0. Este proceso es utilizado en la sección 5.4 para el análisis de resultados.

5.4 ANÁLISIS DE RESULTADOS

Al cumplir con la ejecución del experimento se obtuvieron los resultados presentados en la tabla 3.

NÚMERO DE IMÁGENES MEJOR VALORADAS	CANTIDAD DE DE IMÁGENES DEVUELTAS	NÚMERO DE IMÁGENES CORRECTAMENTE RECUPERADAS	PRECISIÓN
5	1000	608	60.8 %
4	800	591	73.875 %
3	600	538	89.67 %
2	400	392	98.00 %
1	200	199	99.50 %

Cuadro 3: Resultados de precisión de acuerdo al número de imágenes mejor valoradas .

Para realizar el análisis de resultados y obtener la precisión y comprobar la efectividad del método SURF, se toma en cuenta las 5 primeras imágenes mejor valoradas en el ranking que a continuación se detalla :

- Al realizar el análisis con 5 imágenes, se obtiene una precisión de 60.8 %, resultando al menos una coincidencia entre el par de imágenes (original y TRS). Como se puede observar en la tabla 3, el método SURF no demuestra ser tan efectivo, puesto que de las 1000 imágenes devueltas, existe una diferencia de 392 que no coinciden con la imagen original, en la figura 37 se puede observar esta ilustración.
- Seleccionando 4 imágenes se obtiene una precisión de 73.875 %, claramente se puede observar que el método sigue dando un porcentaje de error, esto se debe, que al momento de aplicar la operación de traslación, rotación y escala de forma simultánea y aleatoria, la imagen resulta en ocasiones muy pequeña imposibilitando el reconocimiento y haciendo que las coincidencias entre la imagen original y la TRS sean poco probables.
- Tomando 3 imágenes, se puede apreciar que el porcentaje de precisión va incrementando a 89.69 % , dando un acierto de 538 imágenes recuperadas correctamente de las 600 devueltas.
- Con 2 imágenes la precisión es bastante aceptable puesto que de 400 imágenes devueltas solamente 8 no coinciden con la original.
- Realizando el análisis con 1 imagen, el experimento resulta efectivo dando aciertos de 199 imágenes recuperadas correctamente de las 200, con una precisión de 99.5 %.

Imagen original	Imagen TRS	Valor de Coincidencias	Puntuacion
0.ppm	R_0.ppm	574	1
0.ppm	T_R_0.ppm	188	1
0.ppm	R_S_152.ppm	68	0
0.ppm	R_199.ppm	66	0
0.ppm	T_R_S_141.ppm	64	0
12.ppm	R_12.ppm	592	1
12.ppm	T_R_12.ppm	254	1
12.ppm	T_S_12.ppm	184	1
12.ppm	R_S_12.ppm	134	1
12.ppm	R_S_152.ppm	98	0

(a) Resultados utilizando 5 imágenes mejor valoradas



(b) Imagen 0.ppm



(c) Imagen T_R_S_141.ppm

Figura 37: Utilizando las 5 imágenes mejor valoradas según los puntos de coincidencias

Se puede concluir que, al realizar una sola operación a la imagen original en nuestro caso la rotación, la precisión como resultado al aplicar el método SURF es muy elevada. Por lo contrario, si se realizan tres operaciones de manera simultanea sobre una imagen, el porcentaje de precisión va a ser menor, debido a que al aplicar las operaciones de rotación, traslación y escala, se pierden las características que hacen identificable a la imagen.

En el gráfico de barras (figura 38) se puede apreciar de mejor manera la efectividad del experimento.

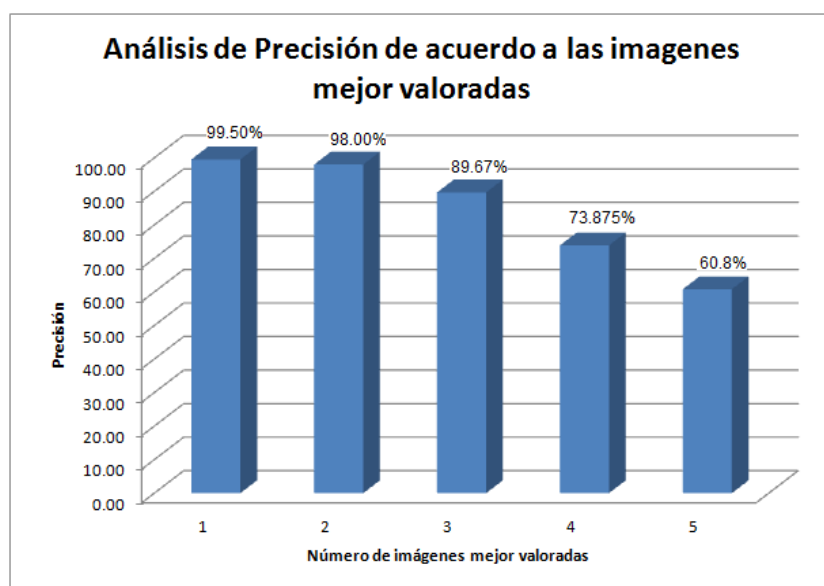


Figura 38: Análisis de precisión del experimento

CONCLUSIONES

Con el trabajo realizado se ha llegado a las siguientes conclusiones:

Se cumplió el objetivo principal de la tesis, siendo este el estudio de métodos de recuperación de imágenes centrándose de manera concreta en el método SURF.

Al realizar un estudio de los métodos SIFT, SURF, Wavelets y sus variantes, se llegó a la conclusión que en muchos de los casos no es factible utilizar solo un método sino la combinación de las mismas ya que unas presentan mejoras con respecto a otras.

De los métodos investigados se pudo verificar que el que mayores prestaciones y rendimiento tiene es SURF, pero que a diferencia de A-SIFT siendo una variante de SIFT, no es capaz de reconocer imágenes donde se cambia el ángulo de la cámara.

En el experimento realizado, cuando se aplican operaciones de rotación, traslación, escala y sus combinaciones con valores aleatorios, no siempre se puede garantizar que las imágenes resultantes preserven sus características de aspecto que las hacen reconocibles.

Al realizar el proceso de extracción de descriptores y comparaciones entre las imágenes originales y las TRS se demoró un tiempo considerablemente largo, tardando 18 horas en procesar 200.000 comparaciones, siendo este motivo más que suficiente para llevar a cabo este proceso solamente con 200 imágenes originales.

En la investigación de las herramientas se llegó a la conclusión de que OpenCV es la mejor, puesto que presenta mayores beneficios teniendo 500 algoritmos para el tratamiento de imágenes, así como librerías y funciones que utilizamos para llevar a cabo nuestro experimento.

Al momento de realizar las pruebas de la herramienta se pudo comprobar que la técnica SURF implementada con OpenCV presenta un buen rendimiento y es capaz de detectar correctamente los puntos característicos de las imágenes originales y en las que se ha insertado ruido.

RECOMENDACIONES

Como trabajo futuro se recomienda realizar una combinación de sistemas de extracción de características, ya que esto permitirá construir sistemas de recuperación de imágenes más robustos.

A fin de realizar una mejor experimentación se sugiere trabajar con un proceso basado en hilos, de esta forma se puede aprovechar de mejor manera el cálculo de descriptores y reducir el tiempo de ejecución.

Una alternativa interesante que se podría probar es la utilización de técnicas de pre - procesamiento de imágenes, a fin de analizar cómo estas afectan en la detección de puntos de interés realizada por SURF. Entre estas técnicas se podría emplear métodos para detectar bordes, filtros o incluso compresión basada en Wavelets.

Se plantea también la posibilidad de que se emplee SURF para el seguimiento de objetos en aplicaciones que utilicen video.

BIBLIOGRAFÍA

- [1] M. Cazorla A. M. Romero. Comparativa de detectores de características visuales y su aplicación al SLAM. Septiembre 2009. (Cited on page 7.)
- [2] Herbert Chuctaya Gabriel Humpire Alexander Oca, Alexander Rodriguez. Reconocimiento de Rostros mediante puntos característicos locales. 2008. (Cited on page 36.)
- [3] Nadia Funes Bernarda Albanesi. Reconocimiento de objetos en video utilizando SIFT Paralelo. Diciembre 2010. (Cited on pages xiii and 13.)
- [4] Óscar Boullosa. Estudio comparativo de descriptores visuales para la detección de escenas Cuasi-Duplicadas. Febrero 2011. (Cited on pages 7, 8, and 19.)
- [5] Maria Fernanda Vilera Suárez Cristina Fernández Jiménez, Paloma Vaquero Quintana. Recuperación de imágenes. (Cited on page 1.)
- [6] Universidad Tecnológica de Pereira. Detección de rostros en imágenes digitales usando clasificadores en cascada. 38, Junio 2008. ISSN 01221701. (Cited on pages xiii and 22.)
- [7] Juan Manuel Peraza Domínguez. Estimación de la distancia recorrida por un robot móvil mediante la utilización de descriptores SURF. Noviembre 2009. (Cited on page 24.)
- [8] Abel Alguacil Gómez. Aplicaciones del operador SIFT al reconocimiento de objetos. Diciembre 2009. (Cited on page 8.)
- [9] Luc Van Gool Herbert Bay, Beat Fasel. Interactive Museum Guide: Fast and Robust Recognition of Museum Objects. . (Cited on page 25.)
- [10] Luc Van Gool Herbert Bay, Tinne Tuytelaars. SURF: Speeded Up Robust Features. . (Cited on pages xiii, 21, 22, 23, and 25.)
- [11] Joaquín López Herraiz. Wavelets: Concepto y aplicaciones para el análisis de señales . Octubre 2004. (Cited on pages xiii, 30, and 38.)
- [12] Pianetti Iván Ing. Herigert Marilina, Bouchet Nahuel. Reconocimiento de imágenes mediante Scale Invariant Feature Transformation (SIFT). (Cited on pages xiii and 9.)
- [13] Dr. Jose Hernandez Palancar Ing. Leonardo Chang Fernandez. Reconocimiento de Rostros usando FPGAs. Octubre 2010. ISSN 2072-6287. (Cited on pages 10 and 24.)
- [14] Ing. Rodolfo A. Cavallero Ing. Walter J. D. Cova. Sobre Wavelets e Imágenes. Noviembre 2006. (Cited on page 31.)
- [15] Txema Franco Iradi. Concepto y características de la imagen. Enero 2004. (Cited on page 51.)

- [16] Guoshen Yu Jean-Michel Morel. ASIFT: A New Framework for Fully Affine Invariant Image Comparison. 2, 2009. (Cited on pages [xiii](#), [10](#), and [11](#).)
- [17] Luis Enrique Palafox Maestre José Jaime Esqueda Elizondo. Fundamentos de procesamiento de imágenes. 2005. (Cited on page [33](#).)
- [18] Wei lin Chao. Gabor wavelet transform and its application. (Cited on page [38](#).)
- [19] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. January 2004. (Cited on pages [7](#), [8](#), and [11](#).)
- [20] Mayka Carretero Rocamora Manuel Agustí i Melchor, Jose Miguel Valiente González. Recuperación por contenido en base de datos de imágenes basada en wavelets: aplicación al diseño del textil. 2003. (Cited on page [36](#).)
- [21] Rubén Orozco Morales Mauricio Orozco Alzate, German Castellanos Domínguez. Extracción de características usando wavelets en la identificación de patologías de ECG. (Cited on page [37](#).)
- [22] Juan Andrade-Cetto Michael Villamizar, Alberto Sanfeliu. Cómputo de características invariantes a la rotación para el reconocimiento de desitintas clases de objetos. 2006. (Cited on page [19](#).)
- [23] Dr. Ing. Jesús Rubén Azor Montoya. Transformada de Fourier y SIFT. (Cited on pages [xiii](#), [32](#), [33](#), [34](#), [35](#), and [36](#).)
- [24] Rubén Mariblanca Nieves. Navegación mediante Visión Estéreo y Slam. Junio 2010. (Cited on page [25](#).)
- [25] Abel Pacheco Ortega. Adecuación de técnicas de descripción Visual utilizando informacion 3D y su aplicación en Robótica. 2011. (Cited on pages [xiii](#), [13](#), [14](#), and [15](#).)
- [26] Cristobal Gómez Barba Óscar Asaustre García. Compresión con wavelet. (Cited on pages [xiii](#) and [32](#).)
- [27] David Otero. Fusión de secuencias de vídeo de alta velocidad procedentes de cámaras desplazadas espacial o temporalmente. Septiembre 2010. (Cited on pages [xiii](#), [19](#), and [21](#).)
- [28] Antonio Nicolás Pina. Clasificación y búsqueda de imágenes usando características visuales. Junio 2010. (Cited on pages [xiii](#), [9](#), [19](#), [24](#), and [25](#).)
- [29] David Leandro Palomino Salcedo. Aplicación de la transformada wavelet continua en el estudio de las secuencias de aminoácidos en una proteína artificial. 2001. URL <http://es.scribd.com/doc/78352081/17/WAVELETS-DAUBECHIES>. (Cited on pages [37](#) and [38](#).)
- [30] Antonio Hernández Sánchez. Sistema Clasificador de Imágenes en Web. Mayo 2005. URL http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/hernandez_s_a/. (Cited on page [29](#).)
- [31] Alfonso Fernández Sarriá. Estudio de técnicas basadas en la transformada wavelet y optimización de sus parámetros para la clasificación por texturas de imágenes digitales. Febrero 2007. (Cited on page [32](#).)

- [32] Eduardo Pedro Serrano. Introducción a la transformada wavelet y sus aplicaciones al procesamiento de señales de emisión acústica. (Cited on pages 31 and 36.)
- [33] Piotr Jasiobedzki Tai-Jin Moyung Stephen Se, Ho-Kong Ng. Vison Based Modeling and localization for planetary exploration rovers. (Cited on page 15.)
- [34] Ing. Gabriel Sánchez Suárez. FPGA. (Cited on page 15.)
- [35] Ing. Luis Vasquez. Tutorial para el uso de imágenes en Octave. URL <http://iie.fing.edu.uy/investigacion/grupos/gti/timag/papers/octima-tutor.pdf>. (Cited on page 51.)
- [36] Rahul Sukthankar Yan Ke. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. (Cited on page 12.)
- [37] Patricio Alejandro Loncomilla Zambrana. Generación automática de Landmarks visuales naturales tridimensionales para Slam Visual. Mayo 2011. (Cited on page 10.)

Part VI

ANEXOS



ANEXOS

A.1 CODIGO SURF

```
/* * A Demo to OpenCV Implementation of SURF * Further Information
    Refer to "SURF: Speed-Up Robust Feature" * Author: Liu Liu *
    liuliu.1987+opencv@gmail.com */
#include "stdafx.h"
#include "opencv2/objdetect/objdetect.hpp"
#include "opencv2/features2d/features2d.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/calib3d/calib3d.hpp"
#include "opencv2/imgproc/imgproc_c.h"
#include <iostream>
using namespace std;
#define USE_FLANN //Es una librería rapida de OpenCV para aproximar
    al vecino más cercano
IplImage* image = 0;
//método de comparación de los puntos descriptores para evaluar la
    correspondencia
double compareSURFDescriptors( const float* d1, const float* d2,
    double best, int length ){
    double total_cost = 0;
    assert( length % 4 == 0 );
    for( int i = 0; i < length; i += 4 ){
        //calculo de las distancias entre la imágenes
        original y TRS
        double t0 = d1[i ] - d2[i ];
        double t1 = d1[i+1] - d2[i+1];
        double t2 = d1[i+2] - d2[i+2];
        double t3 = d1[i+3] - d2[i+3];
        total_cost += t0*t0 + t1*t1 + t2*t2 + t3*t3;
        if( total_cost > best )
            break;
    }
    return total_cost;
}
//método que devuelve a los vecinos mas cercanos
int naiveNearestNeighbor(const float* vec, int laplacian,const
    CvSeq* model_keypoints,const CvSeq* model_descriptors ){
    int length = (int)(model_descriptors->elem_size/sizeof(
        float));
    int i, neighbor = -1;
    double d, dist1 = 1e6, dist2 = 1e6;
    CvSeqReader reader, kreader;//secuencia para la lectura de
        la imagen TRS
    cvStartReadSeq( model_keypoints, &kreader, 0 );//lectura de
        los puntos claves de la imagen TRS
    cvStartReadSeq( model_descriptors, &reader, 0 );//lectura
        de los descriptores de la imagen TRS
    for( i = 0; i < model_descriptors->total; i++ ){
        // Avanza a través de puntos clave/descriptores
        const CvSURFPoint* kp = (const CvSURFPoint*)kreader
            .ptr;
```

```

const float* mvec = (const float*)reader.ptr;
CV_NEXT_SEQ_ELEM( kreader.seq->elem_size, kreader )
    ;//se desplaza hacia el siguiente elemento de
    los puntos claves según su tamaño y la
    secuencia de lectura
CV_NEXT_SEQ_ELEM( reader.seq->elem_size, reader )
    ;//se desplaza hacia el siguiente elemento de
    los descriptores
if( laplacian != kp->laplacian )
    continue;
d = compareSURFDescriptors( vec, mvec, dist2,
    length );//compara las distancias entre los
    descriptores de la imagen original y la imagen
    TRS
if( d < dist1 ){
    dist2 = dist1;
    dist1 = d;
    neighbor = i;
}
else if ( d < dist2 )
    dist2 = d;
}
if ( dist1 < 0.6*dist2 )
    return neighbor;
return -1;
}
//método que encuentra los pares de puntos entre las dos imágenes
void findPairs( const CvSeq* objectKeypoints, const CvSeq*
    objectDescriptors, const CvSeq* imageKeypoints, const CvSeq*
    imageDescriptors, vector<int>& ptpairs ){
    int i;
    // Crear las estructuras que nos ayudan a iterar a través
    de puntos claves/descriptores
    CvSeqReader reader, kreader;//secuencia para la lectura de
    la imagen TRS
    cvStartReadSeq(objectKeypoints, &kreader );//lectura de los
    puntos claves de la imagen original
    cvStartReadSeq( objectDescriptors, &reader );//lectura de
    los descriptores de la imagen original
    ptpairs.clear();
    for( i = 0; i < objectDescriptors->total; i++ ){
        // Avanza a través de puntos clave de objeto/
        descriptores
        const CvSURFPoint* kp = (const CvSURFPoint*)kreader
            .ptr;
        const float* descriptor = (const float*)reader.ptr;
        CV_NEXT_SEQ_ELEM( kreader.seq->elem_size, kreader )
            ;//se desplaza hacia el siguiente elemento de
            los puntos claves
        CV_NEXT_SEQ_ELEM( reader.seq->elem_size, reader )
            ;//se desplaza hacia el siguiente elemento de
            los descriptores
        //busca el vecino mas cercano en la imagen
        int nearest_neighbor=naiveNearestNeighbor(
            descriptor,kp->laplacian,imageKeypoints,
            imageDescriptors);
        if( nearest_neighbor >= 0 ){
            ptpairs.push_back(i);
            ptpairs.push_back(nearest_neighbor);
        }
    }
}

```

```

    }
}
}
//Busca los pares de puntos correspondiente entre las dos imágenes
void flannFindPairs(const CvSeq*,const CvSeq* objectDescriptors,
    const CvSeq*,const CvSeq* imageDescriptors,vector<int>& ptpairs
){
    int length = (int)(objectDescriptors->elem_size/sizeof(
        float));
    cv::Mat m_object(objectDescriptors->total, length, CV_32F)
        ;//Matriz para guardar los descriptores de la imagen
        original
    cv::Mat m_image(imageDescriptors->total, length, CV_32F);//
        Matriz para guardar los descriptores de la imagen TRS
    CvSeqReader obj_reader;//secuencia para la lectura de la
        imagen original
    float* obj_ptr = m_object.ptr<float>(0);
    cvStartReadSeq( objectDescriptors, &obj_reader );//lectura
        de los puntos claves de la imagen original
    for(int i = 0; i < objectDescriptors->total; i++){
        const float* descriptor = (const float*)obj_reader.
            ptr;
        CV_NEXT_SEQ_ELEM( obj_reader.seq->elem_size, obj_
            reader );
        memcpy(obj_ptr, descriptor, length*sizeof(float))
            ;//copia los descriptores de la imagen original
            a la variable obj_ptr
        obj_ptr += length;
    }
    CvSeqReader img_reader;//secuencia para la lectura de la
        imagen TRS
    float* img_ptr = m_image.ptr<float>(0);
    cvStartReadSeq( imageDescriptors, &img_reader );//lectura
        de los puntos claves de la imagen TRS
    for(int i = 0; i < imageDescriptors->total; i++){
        const float* descriptor = (const float*)img_reader.
            ptr;
        CV_NEXT_SEQ_ELEM( img_reader.seq->elem_size, img_
            reader );
        memcpy(img_ptr, descriptor, length*sizeof(float))
            ;//copia los descriptores de la imagen TRS a la
            variable img_ptr
        img_ptr += length;
    }
    // encuentra el vecino más cercano usando FLANN
    cv::Mat m_indices(objectDescriptors->total, 2, CV_32S);
    cv::Mat m_dists(objectDescriptors->total, 2, CV_32F);
    cv::flann::Index flann_index(m_image,cv::flann::
        KDTreeIndexParams(4));//usando 4 kd-arboles aleatorios
    flann_index.knnSearch(m_object,m_indices,m_dists, 2,cv::
        flann::SearchParams(64));//maximo numero de hojas
        comprobados
    int* indices_ptr = m_indices.ptr<int>(0);
    float* dists_ptr = m_dists.ptr<float>(0);
    for (int i=0;i<m_indices.rows;++i) {
        if (dists_ptr[2*i]<0.6*dists_ptr[2*i+1]){
            ptpairs.push_back(i);
            ptpairs.push_back(indices_ptr[2*i]);
        }
    }
}

```

```

    }
}
/*método que busca los puntos de interés y descriptores del objeto
(objectKeypoints,objectDescriptors) y de la imagen (imageKeypoints,
imageDescriptors). En caso de que se encuentren coincidencias
guarda sus coordenadas en la variable dst_corners*/
int locatePlanarObject( const CvSeq* objectKeypoints, const CvSeq*
objectDescriptors,const CvSeq* imageKeypoints, const CvSeq*
imageDescriptors, const CvPoint src_corners[4], CvPoint dst_
corners[4]){
    double h[9];
    CvMat _h = cvMat(3, 3, CV_64F, h);
    vector<int> ptpairs;
    vector<CvPoint2D32f> pt1, pt2;
    CvMat _pt1, _pt2;
    int i, n;
    //encontrar los pares de coincidencias entre las dos
    imágenes original y TRS
#ifdef USE_FLANN
        flannFindPairs( objectKeypoints, objectDescriptors,
            imageKeypoints, imageDescriptors, ptpairs );
#else
        findPairs(objectKeypoints,objectDescriptors,
            imageKeypoints, imageDescriptors, ptpairs );
#endif
    n = (int)(ptpairs.size()/2);
    if( n < 4 )
        return 0;
    pt1.resize(n);
    pt2.resize(n);
    for( i = 0; i < n; i++){
        pt1[i]=((CvSURFPoint*)cvGetSeqElem(objectKeypoints,
            ptpairs[i*2]))->pt;
        pt2[i]=((CvSURFPoint*)cvGetSeqElem(imageKeypoints,
            ptpairs[i*2+1]))->pt;
    }
    _pt1 = cvMat(1, n, CV_32FC2, &pt1[0] );
    _pt2 = cvMat(1, n, CV_32FC2, &pt2[0] );
    //Validacion, si la transformacion entre los puntos de la
    imagen original y TRS son diferentes
    if(!cvFindHomography(&_pt1,&_pt2,&_h,CV_RANSAC,5))
        return 0;
    for( i = 0; i < 4; i++){
        double x = src_corners[i].x, y = src_corners[i].y;
        double Z = 1./(h[6]*x + h[7]*y + h[8]);
        double X = (h[0]*x + h[1]*y + h[2])*Z;
        double Y = (h[3]*x + h[4]*y + h[5])*Z;
        dst_corners[i] = cvPoint(cvRound(X), cvRound(Y));
    }
    return 1;
}
//Metodo principal en el cual se envian las imágenes a ser
comparadas
int main(int argc, char** argv){
    //imagen original
    const char* object_filename = argc == 3 ? argv[1] : "
    paisaje.pgm";
    //imagen a comparar

```

```

const char* scene_filename = argc == 3 ? argv[2] : "paisaje
1.pgm";
//transformar a escala de grises
IplImage* object = cvLoadImage( object_filename, CV_LOAD_
IMAGE_GRAYSCALE );//lee la imagen original forzando a
convertirle a escala de grises
IplImage* image = cvLoadImage( scene_filename, CV_LOAD_
IMAGE_GRAYSCALE );//lee la imagen TRS forzando a
convertirle a escala de grises
IplImage *lectura=cvLoadImage(object_filename,0);
if( !object || !image ){
    fprintf( stderr, "Can not load %s and/or %s\n",
object_filename, scene_filename );
    exit(-1);
}
//almacenamiento de memoria para los puntos característicos
CvMemStorage* storage = cvCreateMemStorage(0);
static CvScalar colors[] =
{
    {{0,0,255}},
    {{0,128,255}},
    {{0,255,255}},
    {{0,255,0}},
    {{255,128,0}},
    {{255,255,0}},
    {{255,0,0}},
    {{255,0,255}},
    {{255,255,255}}
};
IplImage* object_color= cvCreateImage(cvGetSize(object), 8,
3);
cvCvtColor( object, object_color, CV_GRAY2BGR );//convertir
la imagen original a escala de grises
CvSeq* objectKeypoints = 0, *objectDescriptors = 0;
CvSeq* imageKeypoints = 0, *imageDescriptors = 0;
int i;
CvSURFParams params = cvSURFParams(500, 1);//parametro en
el cual se especifica el valor de la hessiana siendo
por defecto 500 y el tamaño de los descriptores 1=128
double tt = (double)cvGetTickCount();
cvExtractSURF(object,0,&objectKeypoints,&objectDescriptors,
storage,params );
cvExtractSURF(image,0,&imageKeypoints,&imageDescriptors,
storage,params );
tt = (double)cvGetTickCount() - tt;
//encuentra las coincidencias entre la imagen original y la
TRS
#ifdef USE_FLANN
#endif
    vector<int> ptpairs;//contiene los puntos de
coincidencia entre las dos imágenes(original y
TRS)
#ifdef USE_FLANN
    flannFindPairs(objectKeypoints,objectDescriptors,
imageKeypoints,imageDescriptors,ptpairs );
#else
    findPairs(objectKeypoints,objectDescriptors,
imageKeypoints, imageDescriptors, ptpairs );
#endif

```

```
        cout << "Puntos correspondientes "<<ptpairs.size()  
            << endl;  
        exit(0);  
    }
```