



INSTITUTO POLITÉCNICO NACIONAL

**ESCUELA SUPERIOR DE
INGENIERÍA MECÁNICA Y ELÉCTRICA**

DETECCIÓN DE BORDES MEDIANTE INTELIGENCIA
ARTIFICIAL

MODALIDAD: CURRICULAR

QUE PARA OBTENER EL TÍTULO:
INGENIERO EN COMUNICACIONES Y
ELECTRÓNICA

PRESENTAN
MORALES GUZMÁN SAÚL
ACOSTA VÁZQUEZ SERGIO ABRAHAM

ASESORES:
DRA. MARÍA ELENA ACEVEDO MOSQUEDA
M. EN C. MARCO ANTONIO ACEVEDO MOSQUEDA

M. EN C. ROBERTO GALICIA GALICIA

MÉXICO D.F 2013



INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELECTRICA
UNIDAD PROFESIONAL “ADOLFO LÓPEZ MATEOS”

TEMA DE TESIS

QUE PARA OBTENER EL TÍTULO DE
POR LA OPCIÓN DE TITULACIÓN
DEBERA(N) DESARROLLAR

INGENIERO EN COMUNICACIONES Y ELECTRÓNICA
TESIS COLECTIVA Y EXAMEN ORAL INDIVIDUAL
C. SAÚL MORALES GUZMÁN
C. SERGIO ACOSTA VÁZQUEZ

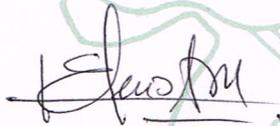
“DETECCIÓN DE BORDES MEDIANTE INTELIGENCIA ARTIFICIAL”

UTILIZAR HERRAMIENTAS DE INTELIGENCIA PARA LA DETECCIÓN DE BORDES EN
IMÁGENES UTILIZANDO EL ENFOQUE ASOCIATIVO.

- INTRODUCCIÓN
- MARCO DE REFERENCIA
- DESARROLLO
- PRUEBAS Y RESULTADOS
- CONCLUSIÓN

MÉXICO D.F. A 06 DE DICIEMBRE DE 2013

ASESORES



DRA. MARÍA ELENA ACEVEDO MOSQUEDA



M. EN C. MARCO ANTONIO ACEVEDO MOSQUEDA



M. EN C. ROBERTO GALICIA GALICIA



ING. PATRICIA LORENA RAMÍREZ RANGEL
JEFE DEL DEPARTAMENTO DE
INGENIERÍA EN COMUNICACIONES Y ELECTRÓNICA



Agradecimientos

DEDICO EL PRESENTE TRABAJO.

A DIOS, por darme la fortaleza
para seguir siempre adelante,
a pesar de los problemas,
por estar conmigo en cada paso de mi vida
y nunca abandonarme.

A mis PADRES y mi FAMILIA
por estar siempre apoyándome
en todo momento, por el sacrificio
de darme una educación de calidad,
para superarme en la vida.

A mis MAESTROS porque a lo largo de este camino,
aprendí todo de ellos, y porque gracias a ellos, logre
llegar hasta donde estoy.

Índice de temas

Agradecimientos	I
Índice de temas	II
Índice de figuras	IV
Resumen	IX
Descripción del problema	X
Justificación	X
Objetivos	XI
Generales.....	XI
Particulares	XI
Solución propuesta para detección de bordes	XI
Capítulo 1 Introducción	1
1.1 Antecedentes en la detección de bordes	1
1.1.1 Métodos basados en la 1ª derivada	2
1.1.2 Métodos basados en la 2ª derivada	9
1.1.3 Algoritmo de Canny.....	14
1.2 Estado del Arte	20
Capítulo 2 Marco de Referencia.....	23
2.1-Conceptos básicos	23
2.2 Memorias asociativas	26
2.2.1 <i>Lernmatrix</i> de Steinbuch.....	26
2.2.2 <i>Correlograph</i> de Willshaw, Buneman y Longuet-Higgins.....	27
2.2.3 <i>Linear Associator</i> de Anderson-Kohonen	28
2.2.4 La memoria asociativa Hopfield	29
2.2.5 Memorias asociativas Morfológicas	31
2.2.6 Memorias asociativas Alfa-Beta.....	33
2.2.7 Memorias Asociativas Media	37
Capítulo 3 Desarrollo.....	39
3.1 Obtención, conversión y segmentación de la imagen.....	40
3.2 Aplicación del algoritmo, de memorias Alfa-Beta	41
3.3 Obtención de eigenvectores	45
3.4 Convolución de máscaras	48
3.5 Selección de eigenvectores	63

3.6 Aplicación de la máscara a la imagen	64
Capítulo 4 Pruebas y Resultados	66
4.1 Eigenvectores a probar	66
4.2 Pruebas para seleccionar eigenvector.....	67
4.3 Resultados finales.....	87
4.4 Comparación de resultados.....	97
Conclusiones	117
Referencias	118
Apéndice A.....	124

Índice de figuras

Fig. 1.1 Cambio brusco de intensidad	1
Fig. 1.2 Tipos de bordes	2
Fig. 1.3 Gráfica del gradiente para imágenes.....	3
Fig. 1.4 Magnitud	3
Fig. 1.5 Orientación	4
Fig. 1.6 Sección de una imagen de 2x2	4
Fig. 1.7 Sección de una imagen de 3x3	5
Fig. 1.8 Máscaras	6
Fig. 1.9 Operador de Roberts	6
Fig. 1.10 Operador de Sobel	6
Fig. 1.11 Operador de Prewitt	7
Fig. 1.12 Imagen seleccionada	7
Fig. 1.13 Aplicando el operador de Roberts	8
Fig. 1.14 Aplicando el operador de Sobel	8
Fig. 1.15 Aplicando el operador de Prewitt	9
Fig. 1.16 Diferencia gráfica entre derivadas de 1º y 2º orden	9
Fig. 1.17 Gráfica del operador LoG 1D.....	10
Fig. 1.18 Aplicación del operador LoG a una imagen.....	11
Fig. 1.19 Gráfica del operador DoG	11
Fig. 1.20 Aplicación del operador DoG	12
Fig. 1.21 Método de la primera derivada.....	13
Fig. 1.22 Método de la segunda derivada	13
Fig. 1.23 Máscaras recomendadas para obtener el filtro gaussiano.....	15
Fig. 1.24 Resultados con el algoritmo de Canny.....	18
Fig. 1.25 Máscara de convolución para cierre de contornos	19
Fig. 1.26 Resultado aplicando el operador de Canny.....	19
Fig. 2.1 Memoria asociativa	23
Fig. 2.2 Fase de aprendizaje Alfa-Beta	33
Fig. 2.3 Fase de recuperación Alfa-Beta	33
Fig. 3.1 Conversión de una imagen RGB a binaria	40
Fig. 3.2 Tomando fragmentos de 3x3, de una imagen de 6x6.....	41
Fig. 3.3 Vector de 9x1.....	41
Fig. 3.4 Vector x vector transpuesto, aplicando la regla Alfa.....	42
Fig. 3.5 Matriz del primer fragmento de la imagen	43
Fig. 3.6 Matriz del segundo fragmento de la imagen	43
Fig. 3.7 Matriz del tercer fragmento de la imagen	44
Fig. 3.8 Matriz del cuarto fragmento de la imagen	44
Fig. 3.9 Obtención de matrices: máximos y mínimos.....	45
Fig. 3.10 Eigenvalores para la matriz de máximos (M).....	46
Fig. 3.11 Eigenvectores para la matriz de máximos (M).....	46
Fig. 3.12 Eigenvalores para la matriz de mínimos (W)	46
Fig. 3.13 Eigenvectores para la matriz de mínimos (W)	47
Fig. 3.14 Primer eigenvector obtenido de la matriz de máximos, convertido a matriz de 3x3	47
Fig. 3.15 Rectángulo	48
Fig. 3.16 Eigenvectores correspondientes a mínimos, aplicados como máscaras al ejemplo 1	49

<i>Fig. 3.17 Eigenvectores correspondientes a máximos, aplicados como máscaras al ejemplo 1</i>	50
<i>Fig. 3.18 Círculo</i>	51
<i>Fig. 3.19 Eigenvectores correspondientes a mínimos, aplicados como máscaras al ejemplo 2</i>	51
<i>Fig. 3.20 Eigenvectores correspondientes a máximos, aplicados como máscaras al ejemplo 2</i>	52
<i>Fig. 3.21 Imagen de Lena</i>	53
<i>Fig. 3.22 Eigenvectores correspondientes a mínimos, aplicados como máscaras al ejemplo 3</i>	53
<i>Fig. 3.23 Eigenvectores correspondientes a máximos, aplicados como máscaras al ejemplo 3</i>	54
<i>Fig. 3.24 Abeja</i>	55
<i>Fig. 3.25 Eigenvectores correspondientes a mínimos, aplicados como máscaras al ejemplo 4</i>	55
<i>Fig. 3.26 Eigenvectores correspondientes a máximos, aplicados como máscaras al ejemplo 4</i>	56
<i>Fig. 3.27 Gato Félix</i>	57
<i>Fig. 3.28 Eigenvectores correspondientes a mínimos, aplicados como máscaras al ejemplo 5</i>	57
<i>Fig. 3.29 Eigenvectores correspondientes a máximos, aplicados como máscaras al ejemplo 5</i>	58
<i>Fig. 3.30 Gato</i>	59
<i>Fig. 3.31 Eigenvectores correspondientes a mínimos, aplicados como máscaras al ejemplo 6</i>	59
<i>Fig. 3.32 Eigenvectores correspondientes a máximos, aplicados como máscaras al ejemplo 6</i>	60
<i>Fig. 3.33 Sombras</i>	60
<i>Fig. 3.34 Eigenvectores correspondientes a mínimos, aplicados como máscaras al ejemplo 7</i>	61
<i>Fig. 3.35 Eigenvectores correspondientes a máximos, aplicados como máscaras al ejemplo 7</i>	61
<i>Fig. 3.36 Dibujo animado</i>	62
<i>Fig. 3.37 Eigenvectores correspondientes a mínimos, aplicados como máscaras al ejemplo 8</i>	62
<i>Fig. 3.38 Eigenvectores correspondientes a máximos, aplicados como máscaras al ejemplo 8</i>	63
<i>Fig. 3.39 Máscaras obtenidas, correspondientes al eigenvector 7</i>	64
<i>Fig. 4.1 Máscaras de Sobel</i>	66
<i>Fig. 4.2 Máscaras de Prewitt</i>	66
<i>Fig. 4.3 Rectángulo de prueba</i>	67
<i>Fig. 4.4 Máscara 4 aplicada al rectángulo</i>	68
<i>Fig. 4.5 Máscara 4 transpuesta aplicada al rectángulo</i>	68
<i>Fig. 4.6 Resultado del rectángulo con eigenvector 4</i>	68
<i>Fig. 4.7 Máscara 6 aplicada al rectángulo</i>	68
<i>Fig. 4.8 Máscara 6 transpuesta aplicada al rectángulo</i>	69
<i>Fig. 4.9 Resultado del rectángulo con eigenvector 6</i>	69
<i>Fig. 4.10 Máscara 7 aplicada al rectángulo</i>	69
<i>Fig. 4.11 Máscara 7 transpuesta aplicada al rectángulo</i>	69

<i>Fig. 4.12 Resultado del rectángulo con eigenvector 7</i>	70
<i>Fig. 4.13 Máscara 8 aplicada al rectángulo</i>	70
<i>Fig. 4.14 Máscara 8 transpuesta aplicada al rectángulo</i>	70
<i>Fig. 4.15 Resultado del rectángulo con eigenvector 8</i>	70
<i>Fig. 4.16 Círculo de prueba</i>	71
<i>Fig. 4.17 Máscara 4 aplicada al círculo</i>	71
<i>Fig. 4.18 Máscara 4 transpuesta aplicada al círculo</i>	71
<i>Fig. 4.19 Resultado del círculo con eigenvector 4</i>	72
<i>Fig. 4.20 Máscara 6 aplicada al círculo</i>	72
<i>Fig. 4.21 Máscara 6 transpuesta aplicada al círculo</i>	72
<i>Fig. 4.22 Resultado del círculo con eigenvector 6</i>	73
<i>Fig. 4.23 Máscara 7 aplicada al círculo</i>	73
<i>Fig. 4.24 Máscara 7 transpuesta aplicada al círculo</i>	73
<i>Fig. 4.25 Resultado del círculo con eigenvector 7</i>	74
<i>Fig. 4.26 Máscara 8 aplicada al círculo</i>	74
<i>Fig. 4.27 Máscara 8 transpuesta aplicada al círculo</i>	74
<i>Fig. 4.28 Resultado del círculo con eigenvector 8</i>	75
<i>Fig. 4.29 Imagen de Lena 2</i>	75
<i>Fig. 4.30 Máscara 4 aplicada a la imagen de Lena 2</i>	76
<i>Fig. 4.31 Máscara 4 transpuesta aplicada a la imagen de Lena 2</i>	76
<i>Fig. 4.32 Resultado de la imagen de Lena 2 con eigenvector 4</i>	77
<i>Fig. 4.33 Máscara 6 aplicada a la imagen de Lena 2</i>	77
<i>Fig. 4.34 Máscara 6 transpuesta aplicada a la imagen de Lena 2</i>	78
<i>Fig. 4.35 Resultado de la imagen de Lena 2 con eigenvector 6</i>	78
<i>Fig. 4.36 Máscara 7 aplicada a la imagen de Lena 2</i>	79
<i>Fig. 4.37 Máscara 7 transpuesta aplicada a la imagen de Lena 2</i>	79
<i>Fig. 4.38 Resultado de la imagen de Lena 2 con eigenvector 7</i>	80
<i>Fig. 4.39 Máscara 8 aplicada a la imagen de Lena 2</i>	80
<i>Fig. 4.40 Máscara 8 transpuesta aplicada a la imagen de Lena 2</i>	81
<i>Fig. 4.41 Resultado de la imagen de Lena 2 con eigenvector 8</i>	81
<i>Fig. 4.42 Siluetas</i>	82
<i>Fig. 4.43 Aplicación de distintas máscaras, a la imagen de siluetas</i>	82
<i>Fig. 4.44 Abeja</i>	83
<i>Fig. 4.45 Aplicación de distintas máscaras, a la imagen de abeja</i>	83
<i>Fig. 4.46 Gato</i>	84
<i>Fig. 4.47 Aplicación de distintas máscaras, a la imagen de gato</i>	84
<i>Fig. 4.48 Félix</i>	85
<i>Fig. 4.49 Aplicación de distintas máscaras, a la imagen de Felix</i>	85
<i>Fig. 4.50 Pingüino</i>	86
<i>Fig. 4.51 Aplicación de distintas máscaras, a la imagen del pingüino</i>	86
<i>Fig. 4.52 Rectángulo</i>	87
<i>Fig. 4.53 Resultado con el rectángulo</i>	87
<i>Fig. 4.54 Círculo</i>	88
<i>Fig. 4.55 Resultado con el círculo</i>	88
<i>Fig. 4.56 Imagen de Lena 3</i>	89
<i>Fig. 4.57 Resultado con la imagen de Lena 3</i>	89
<i>Fig. 4.58 Letra A</i>	90
<i>Fig. 4.59 Resultado con la letra A</i>	90
<i>Fig. 4.60 Letra S</i>	91
<i>Fig. 4.61 Resultado con la letra S</i>	91

<i>Fig 4. 62 Letra C</i>	<i>92</i>
<i>Fig 4. 63 Resultado con la letra C</i>	<i>92</i>
<i>Fig 4. 64 Abeja 2.....</i>	<i>93</i>
<i>Fig 4. 65 Resultado con la abeja 2</i>	<i>93</i>
<i>Fig 4. 66 Siluetas 2.....</i>	<i>94</i>
<i>Fig 4. 67 Resultado con siluetas 2.....</i>	<i>94</i>
<i>Fig 4. 68 Félix 2</i>	<i>95</i>
<i>Fig 4. 69 Resultado con Felix 2.....</i>	<i>95</i>
<i>Fig 4. 70 Escorpión</i>	<i>96</i>
<i>Fig 4. 71 Resultado con escorpión.....</i>	<i>96</i>
<i>Fig. 4.72 Imagen seleccionada 1</i>	<i>97</i>
<i>Fig. 4.73 Resultado 1 obtenido con el método de Prewitt</i>	<i>97</i>
<i>Fig. 4.74 Resultado 1 obtenido con el método de Robert.....</i>	<i>98</i>
<i>Fig. 4.75 Resultado 1 obtenido con el método de Sobel</i>	<i>98</i>
<i>Fig. 4.76 Resultado 1 obtenido con el método de Canny</i>	<i>99</i>
<i>Fig. 4.77 Resultado 1 obtenido con el método propuesto, utilizando las memorias asociativas Alfa-Beta.....</i>	<i>99</i>
<i>Fig. 4.78 Imagen seleccionada 2</i>	<i>100</i>
<i>Fig. 4.79 Resultado 2 obtenido con el método de Prewitt</i>	<i>100</i>
<i>Fig. 4.80 Resultado 2 obtenido con el método de Robert.....</i>	<i>101</i>
<i>Fig. 4.81 Resultado 2 obtenido con el método de Sobel</i>	<i>101</i>
<i>Fig. 4.82 Resultado 2 obtenido con el método de Canny</i>	<i>102</i>
<i>Fig. 4.83 Resultado 2 obtenido con el método propuesto, utilizando las memorias asociativas Alfa-Beta.....</i>	<i>102</i>
<i>Fig. 4.84 Imagen seleccionada 3</i>	<i>103</i>
<i>Fig. 4.85 Resultado 3 obtenido con el método de Prewitt</i>	<i>103</i>
<i>Fig. 4.86 Resultado 3 obtenido con el método de Robert.....</i>	<i>103</i>
<i>Fig. 4.87 Resultado 3 obtenido con el método de Sobel</i>	<i>104</i>
<i>Fig. 4.88 Resultado 3 obtenido con el método de Canny</i>	<i>104</i>
<i>Fig. 4.89 Resultado 3 obtenido con el método propuesto, utilizando las memorias asociativas Alfa-Beta.....</i>	<i>104</i>
<i>Fig. 4.90 Imagen seleccionada 4</i>	<i>105</i>
<i>Fig. 4.91 Resultado 4 obtenido con el método de Prewitt</i>	<i>105</i>
<i>Fig. 4.92 Resultado 4 obtenido con el método de Robert.....</i>	<i>105</i>
<i>Fig. 4.93 Resultado 4 obtenido con el método de Sobel</i>	<i>106</i>
<i>Fig. 4.94 Resultado 4 obtenido con el método de Canny</i>	<i>106</i>
<i>Fig. 4.95 Resultado 4 obtenido con el método propuesto, utilizando las memorias asociativas Alfa-Beta.....</i>	<i>106</i>
<i>Fig. 4.96 Imagen seleccionada 5</i>	<i>107</i>
<i>Fig. 4.97 Resultado 5 obtenido con el método de Prewitt</i>	<i>107</i>
<i>Fig. 4.98 Resultado 5 obtenido con el método de Robert.....</i>	<i>107</i>
<i>Fig. 4.99 Resultado 5 obtenido con el método de Sobel</i>	<i>108</i>
<i>Fig. 4.100 Resultado 5 obtenido con el método de Canny</i>	<i>108</i>
<i>Fig. 4.101 Resultado 5 obtenido con el método propuesto, utilizando las memorias asociativas Alfa-Beta.....</i>	<i>108</i>
<i>Fig. 4.102 Imagen seleccionada 6.....</i>	<i>109</i>
<i>Fig. 4.103 Resultado 6 obtenido con el método de Prewitt</i>	<i>109</i>
<i>Fig. 4.104 Resultado 6 obtenido con el método de Robert.....</i>	<i>110</i>
<i>Fig. 4.105 Resultado 6 obtenido con el método de Sobel</i>	<i>110</i>
<i>Fig. 4.106 Resultado 6 obtenido con el método de Canny</i>	<i>111</i>

<i>Fig. 4.107 Resultado 6 obtenido con el método propuesto, utilizando las memorias asociativas Alfa-Beta</i>	111
<i>Fig. 4.108 Imagen seleccionada 7</i>	112
<i>Fig. 4.109 Resultado 7 obtenido con el método de Prewitt</i>	112
<i>Fig. 4.110 Resultado 7 obtenido con el método de Robert</i>	112
<i>Fig. 4.111 Resultado 7 obtenido con el método de Sobel</i>	113
<i>Fig. 4.112 Resultado 7 obtenido con el método de Canny</i>	113
<i>Fig. 4.113 Resultado 7 obtenido con el método propuesto, utilizando las memorias asociativas Alfa-Beta</i>	113
<i>Fig. 4.114 Imagen seleccionada 8</i>	114
<i>Fig. 4.115 Resultado 8 obtenido con el método de Prewitt</i>	114
<i>Fig. 4.116 Resultado 8 obtenido con el método de Robert</i>	115
<i>Fig. 4.117 Resultado 8 obtenido con el método de Sobel</i>	115
<i>Fig. 4.118 Resultado 8 obtenido con el método de Canny</i>	116
<i>Fig. 4.119 Resultado 8 obtenido con el método propuesto, utilizando las memorias asociativas Alfa-Beta</i>	116

Resumen

El objetivo de esta propuesta es, la detección de bordes mediante memorias asociativas Alfa-Beta. Este problema se resuelve mediante el estudio de las memorias asociativas, y algoritmos ya desarrollados para la detección de bordes.

Primeramente, se debe leer la imagen, una vez leída, ésta será convertida a binaria, es decir, tendrá valores de "1" y "0", esto es porque el algoritmo que se pretende implementar, correspondiente a las memorias asociativas Alfa-Beta, las cuales trabajan únicamente con valores binarios.

Para el siguiente paso, se le hará un pre-procesamiento a la imagen, que consiste básicamente en hacerla múltiplo de 3, si la imagen leída es múltiplo de 3, este paso se omitirá.

Enseguida, se procede a fraccionar la imagen en matrices de 3x3, para posteriormente convertirlos a forma de vector, de un tamaño de 9x1.

Ya finalizado este paso, hacemos uso del algoritmo de las memorias asociativas Alfa-Beta, tomando como patrones de entrada estos vectores de 9x1. Enseguida, nos damos a la tarea, de operar los vectores como patrones de entrada, con el operador α , aplicando la fase de aprendizaje de las memorias Alfa-Beta (figura 27), operamos estos patrones por sí mismos, el primero teniendo una forma de 9x1, y el segundo transpuesto, es decir teniendo una forma de 1x9, resultando así una matriz de 9x9, y por lo tanto, obtendremos un número n matrices de 9x9, dependiendo del tamaño de la imagen.

Ya concluida esta fase, aplicamos el operador de mínimos y máximos, para todas las matrices resultantes, teniendo así dos matrices: una que contendrá el valor más pequeño comparando cada matriz, y otra que contendrá el valor más alto comparando cada matriz.

Después para estas matrices resultantes, procedemos a obtener sus eigenvectores, que servirán como máscaras, para su posterior implementación a la imagen, seleccionando así el más adecuado para obtener un mejor resultado.

Los algoritmos para detección de bordes ya existentes, como son el de: Robert, Prewitt, Sobel Canny, se basan en derivadas. Esta propuesta se basa en memorias asociativas.

Este método propuesto, obtiene resultados comparables, con los métodos ya existentes para detección de bordes, ampliando así el campo de desarrollo de las memorias asociativas.

Descripción del problema

La detección de bordes es un proceso en el análisis digital de imágenes, que detecta los cambios en la intensidad de luz. Estos cambios se pueden usar para determinar profundidad, tamaño, orientación y propiedades de la superficie dentro de una muestra o pieza de trabajo.

Es una técnica de análisis digital que se puede aplicar a imágenes de muestras 2-D y 3-D. Un borde puede ser llamado, como la frontera entre dos regiones diferentes en una imagen. Un borde puede ser el resultado de cambios en la absorción de la luz (color/sombra), textura, etc. Por ejemplo, si una serie lineal de píxeles graban la intensidad de luz de : 2, 3, 2, 4, 3, 2, 95, 97, 96, se espera un borde o discontinuidad entre los píxeles grabados con intensidad de 2 y 95. Estos píxeles pueden ser llamados como "puntos de borde".

La detección de cambios sutiles puede ser confundida por "ruido", y es dependiente del umbral en el píxel de cambio que define un borde [1].

La detección de bordes, a lo largo de los años ha tenido grandes avances con algoritmos, como: Robert, Sobel, Prewitt, Canny, etc., pero estos algoritmos tienen la particularidad, que se basan en derivadas, y por consecuencia, las imágenes resultantes después de aplicar el algoritmo amplifican el ruido que hay en ellas, por lo que, es necesario aplicar un suavizado a la imagen antes de ser procesada, para así atenuar el ruido que hay en ellas, pero esto presenta problemas a la hora de tratar de recuperar información, porque el suavizado de una imagen elimina puntos que posiblemente pueden ser bordes, por ejemplo, de un modo más estricto, esto podría presentar un problema a la hora de tratar de realizar un estudio, donde lo que queremos encontrar, es alguna especie de virus o bacterias y queremos clasificarlas, con los algoritmos existentes, se presenta el problema de que no resaltan algunos puntos de borde, porque en el suavizado se pierden, y si por ejemplo, estos puntos que son eliminados llegasen a ser parte de las bacterias o virus, cabría la posibilidad de tener una mala clasificación.

Justificación

La detección de bordes efectiva ha sido un paso importante para la visión de máquinas, sistemas de procesamiento automático, en la etapa final de procesamiento para un reconocimiento de objetos de alto nivel y para los sistemas de interpretación. Es un proceso fundamental en la industria de la metrología, ya que define las fronteras de la característica a ser medida. La precisión en la detección del borde mejora la precisión en los procesos en curso y los procedimientos de control de calidad [1].

Objetivos

Generales

Utilizar herramientas de inteligencia artificial para la detección de bordes en imágenes utilizando el enfoque asociativo.

Particulares

Estudio de las memorias asociativas.

Implementación de los algoritmos de memorias asociativas.

Aplicación de las memorias asociativas a la detección de bordes.

Comparación de los resultados obtenidos con otros algoritmos de detección de bordes.

Solución propuesta para detección de bordes

En este trabajo se propone, un nuevo método mediante el uso de herramientas de Inteligencia Artificial, como lo son los Modelos Asociativos, para realizar la detección de bordes.

El primer paso, es la implementación de algoritmos de modelos asociativos. Después se implementarán estos algoritmos a la detección de bordes. Se deben estudiar y utilizar (funciones de MatLab, c++), otros algoritmos que realicen la misma tarea para llevar a cabo una comparación, que nos permita evaluar el desempeño de nuestra propuesta.

Los modelos basados en el concepto de memorias asociativas, se han desarrollado y estudiado desde los años sesenta. Una de las aplicaciones que se les puede dar es, para la detección de bordes.

Las memorias asociativas han tenido una gran aceptación, por la efectividad que han demostrado tener, en la clasificación de las diferentes bases de datos.

Capítulo 1 Introducción

La detección de bordes, es un proceso que se aplica, en análisis digital de imágenes, para detectar los cambios bruscos de intensidad entre píxeles vecinos.

Los bordes en una imagen, se pueden definir como transiciones entre dos regiones. Suministran una valiosa información, sobre las fronteras de los objetos, esta información puede ser utilizada para reconocer objetos.

La manera más común para detectar bordes es utilizar algún tipo de derivada o diferencial. La derivada nos permite calcular las variaciones en un punto de la imagen. Viendo la imagen como una función, un contorno implica una discontinuidad en dicha función, es decir, donde la función tiene un valor de gradiente o derivada [2].

En la Fig. 1.1, se muestra una discontinuidad en intensidad entre la parte izquierda y derecha de la imagen.

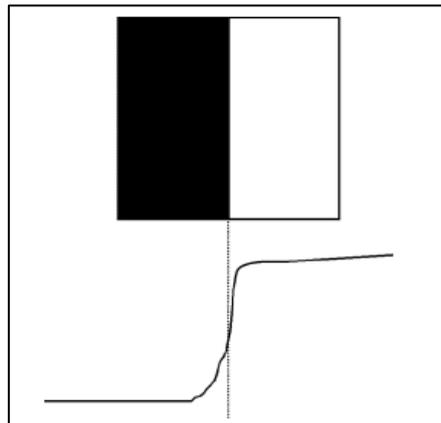


Fig. 1.1 Cambio brusco de intensidad

1.1 Antecedentes en la detección de bordes

La detección de bordes en una imagen reduce significativamente la cantidad de información, y filtra la información innecesaria, preservando las características fundamentales de la imagen.

Podemos definir como un borde a los píxeles, donde la intensidad de la imagen cambia de forma abrupta, los ejes o bordes de una imagen, se van detectando de estas zonas, donde el nivel de intensidad va cambiando bruscamente, cuanto más rápido se produce el cambio de intensidad, el eje o borde es más fuerte.

Para poder detectar los bordes de una imagen, debemos detectar aquellos *puntos borde* que los forman. Así, un punto de borde puede ser visto como un punto en una imagen (píxel), donde se produce una discontinuidad en el gradiente (vector) de la imagen [3], en la Fig. 1.2 se pueden ver los diferentes tipos de bordes que se pueden presentar.

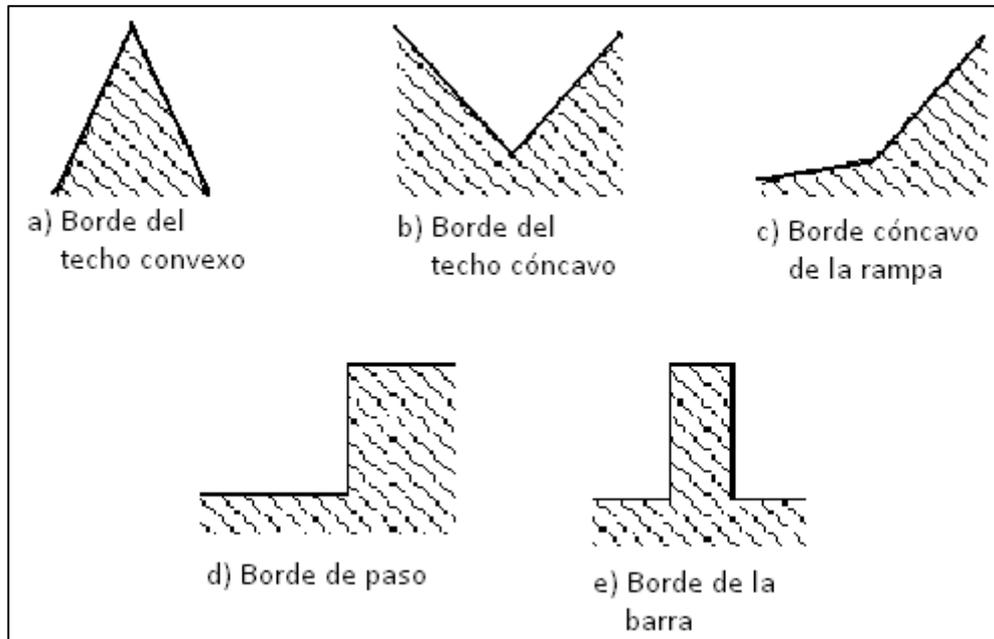


Fig. 1.2 Tipos de bordes

De la Fig. 1.2, podemos clasificar los tipos de bordes en 2 tipos más generales:

- 1.-Versiones suavizadas, (incisos a, b y c).
- 2.-Discontinuidades en la intensidad, (incisos d y e).

1.1.1 Métodos basados en la 1ª derivada

El gradiente es un vector, en donde sus componentes miden la rapidez con que los valores de los píxeles cambian en la distancia, en las direcciones **X** y **Y** [3].

Las técnicas clásicas de detección de bordes se basan en diferenciar a la imagen, esto es, encontrar la derivada respecto a los ejes **X** y **Y**, o con el gradiente. El gradiente de una función **f (x; y)** se define como la siguiente ecuación (1). [2]

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right) \quad (1)$$

La magnitud del gradiente (∇f) se calcula por medio de la ecuación (2).

$$|\nabla f| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \quad (2)$$

En el caso de imágenes discretas, se puede considerar dx y dy en términos del número de píxeles entre dos puntos. Así, cuando $dx=dy=1$ y el punto donde vamos a medir el gradiente, tiene coordenadas (i,j) , tenemos que la siguiente ecuación (3), [3, 4]:

-Gradiente=
1ª derivada de f

$$\nabla f(x,y) = \frac{\partial f}{\partial x} i + \frac{\partial f}{\partial y} j \quad (3)$$

-El gradiente para una imagen es un vector de 2 componentes, como se muestra en la Fig. 1.3.

- Gradiente en X, f_x
- Gradiente en Y, f_y

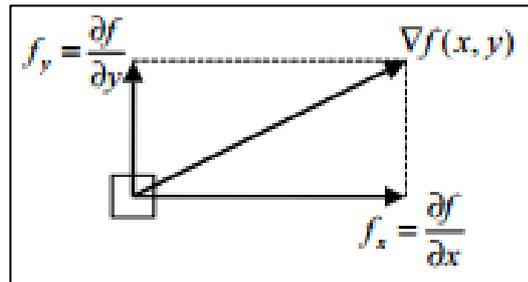


Fig. 1.3 Gráfica del gradiente para imágenes

La magnitud se calcula mediante la ecuación (4), como resultado podemos ver la Fig. 1.4, mostrada a continuación.

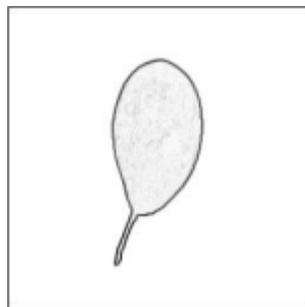


Fig. 1.4 Magnitud

$$|\nabla f(x,y)| = m(x,y) = \sqrt{f_x^2 + f_y^2} \quad (4)$$

La magnitud es igual a la fortaleza del borde.

La orientación se calcula mediante la ecuación (5), y como resultado podemos ver la Fig. 1.5, mostrada a continuación.

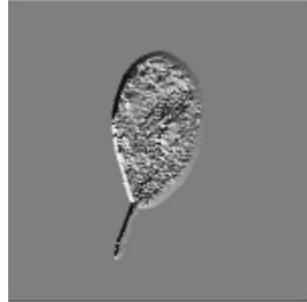


Fig. 1.5 Orientación

$$|\nabla f(x, y)| = \phi(x, y) = \arctan \frac{f_y}{f_x} \quad (5)$$

La orientación es igual a la dirección del borde.

En el caso discreto, podemos aproximar la derivada tomando simplemente la diferencia entre dos valores contiguos. Si consideramos una sección de **2x2** de la imagen como se ve en la Fig. 1.6 [2].

$I_{1,1}$	$I_{1,2}$
$I_{2,1}$	$I_{2,2}$

Fig. 1.6 Sección de una imagen de 2x2

Entonces, una posible aproximación discreta al gradiente en dicha región, es como se muestra en las siguientes ecuaciones (6) y (7):

$$\frac{\partial f}{\partial x} = I_{1,2} - I_{1,1} \quad (6)$$

$$\frac{\partial f}{\partial y} = I_{2,1} - I_{1,1} \quad (7)$$

Donde (6) es el gradiente horizontal y (7) es el gradiente vertical. También podemos extender esta aproximación a un área de la imagen de **3 x 3**, como se muestra en la Fig. 1.7.

$I_{1,1}$	$I_{1,2}$	$I_{1,3}$
$I_{2,1}$	$I_{2,2}$	$I_{2,3}$
$I_{3,1}$	$I_{3,2}$	$I_{3,3}$

Fig. 1.7 Sección de una imagen de 3x3

Aproximando el gradiente en este caso, tenemos la ecuación (8):

$$\frac{\partial f}{\partial x} = (I_{3,1} + I_{3,2} + I_{3,3}) - (I_{1,1} + I_{1,2} + I_{1,3}) \quad (8)$$

$$\frac{\partial f}{\partial y} = (I_{1,3} + I_{2,3} + I_{3,3}) - (I_{1,1} + I_{2,1} + I_{3,1})$$

En el caso bidimensional discreto, las distintas aproximaciones del operador gradiente, se basan en diferencias entre los niveles de grises de la imagen. La derivada parcial **f_x(x,y)** (gradiente de fila **(i,j)**), puede aproximarse por la diferencia de píxeles adyacentes de la misma fila, como se muestra en la ecuación (9).

$$\frac{\partial f(x,y)}{\partial x} \approx \nabla_x f(x,y) = f(x,y) - f(x-1,y) \quad \begin{array}{|c|c|} \hline -1 & 1 \\ \hline \end{array} \quad (9)$$

La discretización del vector gradiente en el eje Y (gradiente de columna **(i,j)**), será, como se muestra en la ecuación (10):

$$\frac{\partial f(x,y)}{\partial y} \approx \nabla_y f(x,y) = f(x,y) - f(x,y-1) \quad \begin{array}{|c|} \hline -1 \\ \hline 1 \\ \hline \end{array} \quad (10)$$

Para la implementación y cálculo del gradiente, se utilizan máscaras o filtros que representan o equivalen a dichas ecuaciones. En este caso, calcular el gradiente sobre toda una imagen con las condiciones de que **dx=dy=1**, consiste en convolucionar la imagen con unas máscaras, como las que se

muestran en la Fig. 1.8, del tipo Δx para detectar bordes verticales, y Δy para detectar bordes horizontales [3].

$$\Delta x = \begin{bmatrix} -1 & 1 \\ 0 & 0 \end{bmatrix} \quad \Delta y = \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix}$$

Fig. 1.8 Máscaras

En vez de determinar el gradiente a lo largo de las direcciones X y Y , también podemos detectarlo en las direcciones de 45° y 135° . En este caso, las máscaras correspondientes se conocen con el nombre de **operador de Roberts**, como se muestran en la Fig. 1.9.

$$\Delta x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad \Delta y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Fig. 1.9 Operador de Roberts

Muchas técnicas basadas en la utilización de máscaras para la detección de bordes, utilizan máscaras de tamaño 3×3 o incluso más grandes.

La ventaja de utilizar máscaras grandes, es que los errores producidos por efectos del ruido, son reducidos mediante medias locales de la matriz, tomadas en los puntos en donde se superpone la máscara. Por otro lado, las máscaras normalmente tienen tamaños impares, de forma que los operadores, se encuentran centrados sobre los puntos en donde se calculan los gradientes.

Otro operador muy conocido es el **operador de Sobel**, en donde las máscaras que se muestran en la Fig. 1.10, buscan ejes en las direcciones horizontales y verticales, y combinan esta información mediante la magnitud.

$$\Delta x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \Delta y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Fig. 1.10 Operador de Sobel

Otro operador que utiliza máscaras 3x3 y es muy parecido al de Sobel, es el de **operador de Prewitt**, diferenciándose únicamente en los coeficientes, como se muestra en la Fig. 1.11 [3].

$$\Delta x = \begin{array}{|c|c|c|} \hline -1 & -1 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \quad \Delta y = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

Fig. 1.11 Operador de Prewitt

A continuación, en la Fig. 1.13 a la Fig. 1.15, se muestran los resultados de aplicar los operadores de: Robert, Sobel y Prewitt, a la imagen seleccionada correspondiente a la Fig. 1.12.

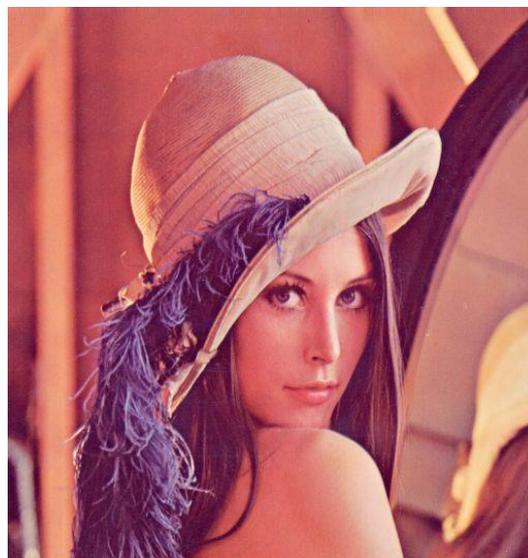


Fig. 1.12 Imagen seleccionada

Aplicando el operador de Robert a la imagen original, tenemos el siguiente resultado, mostrado en la Fig. 1.13.



Fig. 1.13 Aplicando el operador de Roberts

Aplicando el operador de Sobel obtenemos la Fig. 1.14.



Fig. 1.14 Aplicando el operador de Sobel

Aplicando el operador de Prewitt a la imagen original, obtenemos el resultado mostrado en la Fig. 1.15.



Fig. 1.15 Aplicando el operador de Prewitt

1.1.2 Métodos basados en la 2ª derivada

Todos los operadores anteriores, tienen una aproximación hacia derivadas de primer orden, sobre el valor de los píxeles en una imagen, en la Fig. 1.16 podemos ver la diferencia gráfica, entre las derivadas de primer y segundo orden [3].

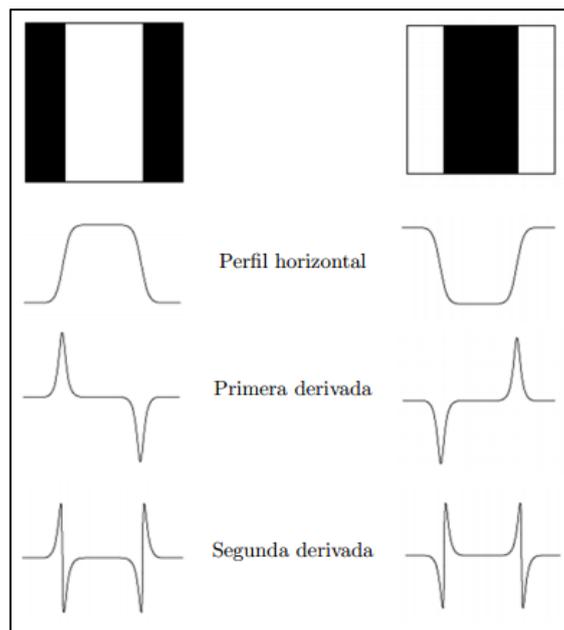


Fig. 1.16 Diferencia gráfica entre derivadas de 1º y 2º orden

Existen métodos que utilizan detectores de bordes basados en derivadas de 2º orden, como se muestra en la ecuación (11). Uno de los más populares es el operador Laplaciano.

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

0	-1	0
-1	4	-1
0	-1	0

(11)

Aunque el Laplaciano responde a las transiciones de intensidad, rara vez se utiliza en la práctica para la detección de bordes:

- Los operadores basados en la primera derivada, son sensibles al ruido en imágenes. El Laplaciano aún lo es más.
- Genera bordes dobles.
- No existe información direccional de los ejes detectados.

Uno de los métodos más utilizados, es el suavizado por medio de una **Gaussiana**.

- Convolucionar la imagen original con un filtro gaussiano.
- Calcular las derivas sobre la imagen suavizada.

Como ambas operaciones son lineales, podemos combinar ambas operaciones de diferentes formas:

- Suavizado de la imagen y cálculo de la 2 derivada (ecuación 12).
- Convolución de la imagen original, utilizando el Laplaciano del Gaussiano (**operador LoG (Laplaciano de la gaussiana)**).

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

↓

$$\nabla^2 G = \left(\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$
(12)

La Fig. 1.17, muestra gráficamente el resultado de aplicar el operador Laplaciano a una Gaussiana, en el caso unidimensional.

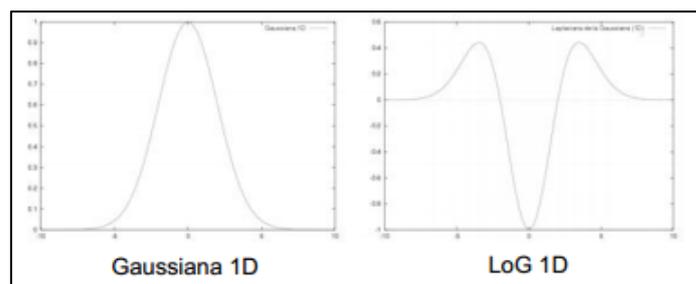


Fig. 1.17 Gráfica del operador LoG 1D

Este método de detección de bordes, fue propuesto por primera vez por Marr and Hildreth, quienes introdujeron el principio de detecciones, mediante el método de cruces por cero.

El principio en que se basa este método, consiste en encontrar las posiciones en una imagen, donde la segunda derivada toma el valor 0.

En la Fig. 1.18 se muestra el resultado de aplicar el operador LoG a una imagen.

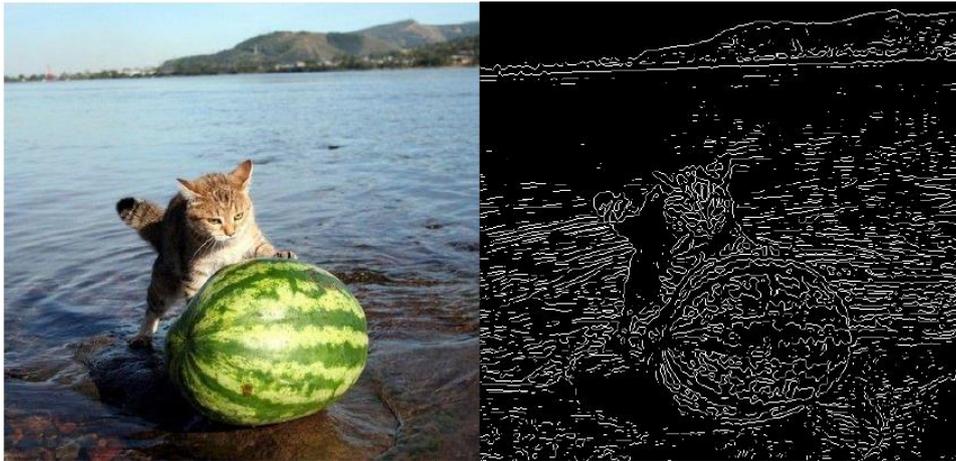


Fig. 1.18 Aplicación del operador LoG a una imagen

Operador DoG

Otro operador que puede aproximarse al operador LoG, es el DoG. Consiste, en tomar la diferencia de dos gaussianas con diferentes desviaciones estándar, como se muestra en la Fig. 1.19. A este operador, se le conoce también como, **operador de Diferencia de Gaussianas** u Operador **de Sombrero Mexicano** [3].

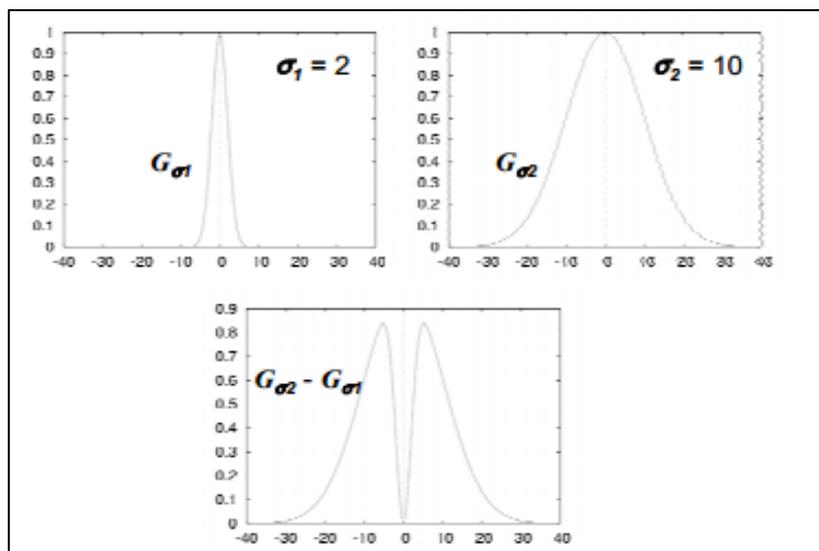


Fig. 1.19 Gráfica del operador DoG

En la Fig. 1.20, se muestra el resultado de aplicar el operador DoG a una imagen; en el resultado obtenido, los colores fueron invertidos, es decir lo que era negro es blanco y lo que es blanco era negro, con el fin de observar de otra manera el resultado.

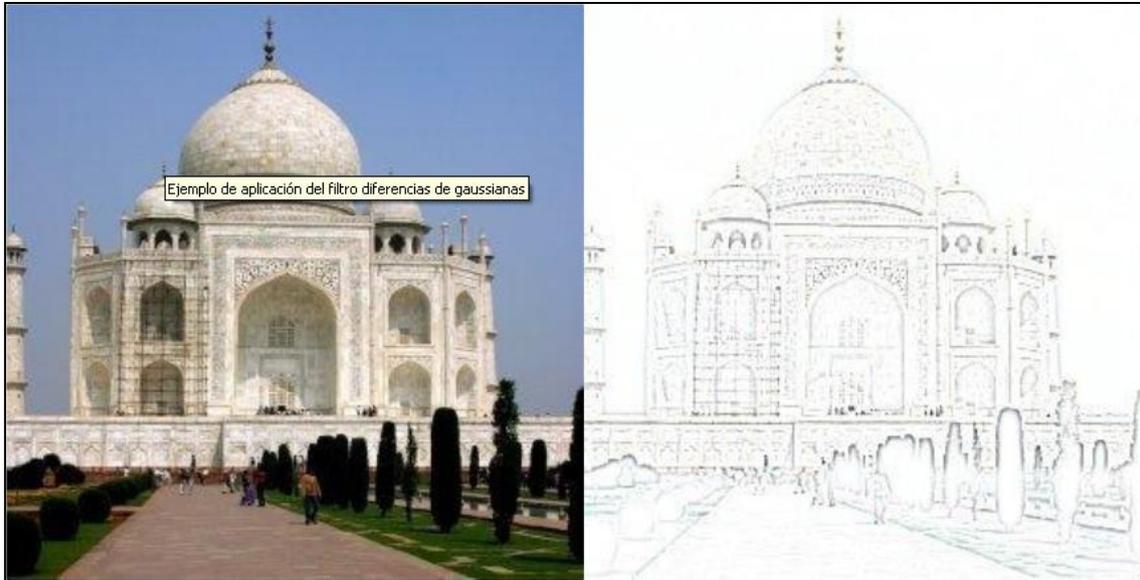


Fig. 1.20 Aplicación del operador DoG

El operador de Marr-Hildreth, llegó a ser uno de los más utilizados por las siguientes razones [3]:

- Sus fundamentos están basados en los campos receptivos de los ojos de animales.
- El operador es simétrico. Los ejes se encuentran en todas las orientaciones, cosa que no sucede con los operadores basados en la primera derivada, los cuales son direccionales.
- Los cruces por cero de la segunda derivada son más fáciles de determinar, que los máximos en la primera derivada. Sólo se necesita detectar un cambio de signo en la señal. Por otro lado, los cruces por cero de una señal, se encuentran siempre sobre contornos cerrados.

Problemas

- La influencia del ruido es considerable en la segunda derivada.
- La generación siempre de contornos cerrados no es realista.
- El operador DoG, marca puntos considerados como ejes en algunas localizaciones donde no hay bordes.

En la Fig. 1.21, se puede ver que la primera derivada, es menos susceptible al ruido, a comparacion de la segunda derivada.

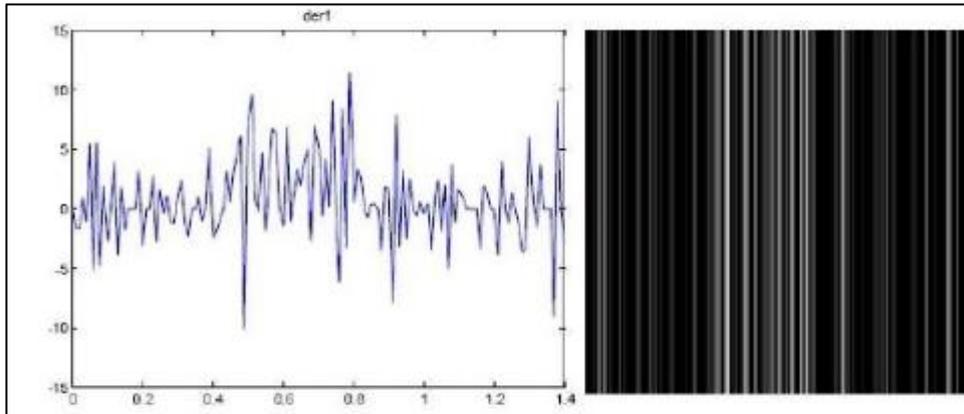


Fig. 1.21 Método de la primera derivada

La Fig. 1.22, representa la segunda derivada, y se puede observar que el ruido es mayor que en la primera derivada, siendo así la segunda derivada más susceptible al ruido.

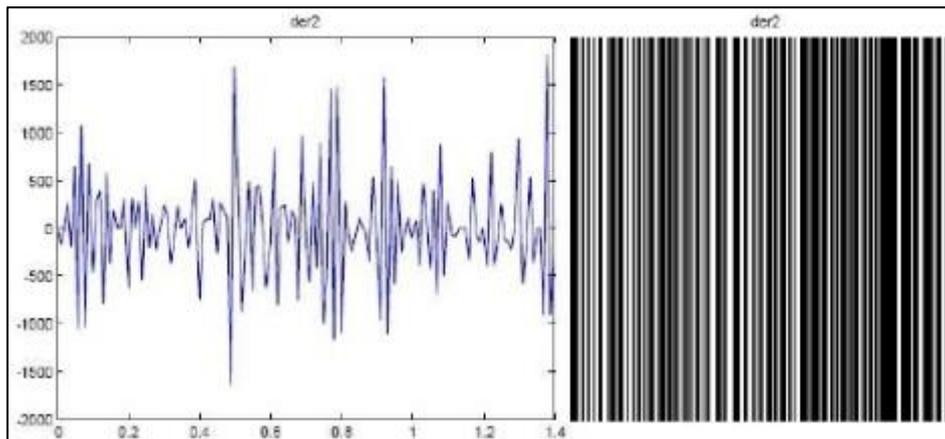


Fig. 1.22 Método de la segunda derivada

Nota.- La segunda derivada es más susceptible al ruido, y por lo tanto, menos utilizada para detectar bordes.

1.1.3 Algoritmo de Canny

En 1986, Canny propuso un método para la detección de bordes, el cual se basaba en tres criterios, éstos son [7]:

- Un criterio de detección, expresa el hecho de evitar la eliminación de bordes importantes y no suministrar falsos bordes.
- El criterio de localización, establece que la distancia entre la posición real y la localizada del borde, se debe minimizar.
- El criterio de una respuesta que integre las respuestas múltiples, correspondientes a un único borde.

Algoritmo de Canny para la detección de bordes

Uno de los métodos relacionados con la detección de bordes, es el uso de la primera derivada, la que es usada por que toma el valor de cero en todas las regiones, donde no varía la intensidad y tiene un valor constante en toda la transición de intensidad. Por tanto, un cambio de intensidad se manifiesta como un cambio brusco en la primera derivada, característica que es usada para detectar un borde, y en la que se basa el algoritmo de Canny [7].

El algoritmo de Canny consiste en tres grandes pasos:

- Obtención del gradiente: en este paso se calcula la magnitud y orientación del vector gradiente en cada píxel.
- Supresión no máxima: en este paso se logra el adelgazamiento del ancho de los bordes, obtenidos con el gradiente, hasta lograr bordes de un píxel de ancho.
- Histéresis de umbral: en este paso se aplica una función de histéresis basada en dos umbrales; con este proceso se pretende reducir la posibilidad de aparición de contornos falsos.

Obtención del gradiente

Para la obtención del gradiente, lo primero que se realiza es la aplicación de un filtro gaussiano a la imagen original, con el objetivo de suavizar la imagen y tratar de eliminar el posible ruido existente. Sin embargo, se debe de tener cuidado de no realizar un suavizado excesivo, pues se podrían perder detalles de la imagen y provocar un pésimo resultado final. Este suavizado se obtiene promediando los valores de intensidad de los píxeles, en el entorno de vecindad con una máscara de consolución de media cero, y desviación estándar σ . En la figura 1.23, se muestran dos ejemplos de máscaras que se pueden usar para realizar el filtrado gaussiano.

En la Fig.1.23, para el inciso **a**, la máscara fue obtenida de la referencia [5], mientras que para el inciso **b**, la máscara fue obtenida de la referencia [6].

(a)	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>1</td><td>4</td><td>7</td><td>4</td><td>1</td></tr> <tr><td>4</td><td>16</td><td>26</td><td>16</td><td>4</td></tr> <tr><td>7</td><td>26</td><td>41</td><td>26</td><td>7</td></tr> <tr><td>4</td><td>16</td><td>26</td><td>16</td><td>4</td></tr> <tr><td>1</td><td>4</td><td>7</td><td>4</td><td>1</td></tr> </table>	1	4	7	4	1	4	16	26	16	4	7	26	41	26	7	4	16	26	16	4	1	4	7	4	1	(b)	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><td>2</td><td>4</td><td>5</td><td>4</td><td>2</td></tr> <tr><td>4</td><td>9</td><td>12</td><td>9</td><td>4</td></tr> <tr><td>5</td><td>12</td><td>15</td><td>12</td><td>5</td></tr> <tr><td>4</td><td>9</td><td>12</td><td>9</td><td>4</td></tr> <tr><td>2</td><td>4</td><td>5</td><td>4</td><td>2</td></tr> </table>	2	4	5	4	2	4	9	12	9	4	5	12	15	12	5	4	9	12	9	4	2	4	5	4	2
1	4	7	4	1																																																	
4	16	26	16	4																																																	
7	26	41	26	7																																																	
4	16	26	16	4																																																	
1	4	7	4	1																																																	
2	4	5	4	2																																																	
4	9	12	9	4																																																	
5	12	15	12	5																																																	
4	9	12	9	4																																																	
2	4	5	4	2																																																	
$\frac{1}{273}$		$\frac{1}{115}$																																																			

Fig. 1.23 Máscaras recomendadas para obtener el filtro gaussiano

Una vez que se suaviza la imagen, para cada píxel se obtiene la magnitud y módulo (orientación) del gradiente. El algoritmo para este primer paso se describe a continuación.

Supresión no máxima al resultado del gradiente

El procedimiento es el siguiente: se consideran cuatro direcciones identificadas por las orientaciones de 0°, 45°, 90° y 135° con respecto al eje horizontal. Para cada píxel se encuentra la dirección que mejor se aproxime a la dirección del ángulo de gradiente.

Algoritmo: Obtención de Gradiente

Entrada: imagen I

Máscara de convolución H , con media cero y desviación estándar σ .

Salida: imagen E_m de la magnitud del gradiente

Imagen E_o de la orientación del gradiente

1. Suavizar la imagen I con H , mediante un filtro gaussiano y obtener J como imagen de salida.
2. Para cada píxel (i, j) en J , obtener la magnitud y orientación del gradiente, basándose en las siguientes expresiones:

El gradiente de una imagen $\mathbf{f}(\mathbf{x},\mathbf{y})$ en un punto (\mathbf{x},\mathbf{y}) , se define como un vector bidimensional dado por la ecuación 13:

$$G[f(x, y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} f(x, y) \\ \frac{\partial}{\partial y} f(x, y) \end{bmatrix} \quad (13)$$

Siendo un vector perpendicular al borde, donde el vector \mathbf{G} apunta en la dirección de variación máxima de \mathbf{f} , en el punto (\mathbf{x},\mathbf{y}) por unidad de distancia, con la magnitud y dirección dadas por las ecuaciones (14) y (15):

$$|G| = \sqrt{G_x^2 + G_y^2} = |G_x| + |G_y| \quad (14)$$

$$\phi(x, y) = \tan^{-1} \frac{G_y}{G_x} \quad (15)$$

3. Obtener E_m a partir de la magnitud de gradiente (14) y E_0 a partir de la orientación (15), de acuerdo a las expresiones anteriores.

Posteriormente se observa, si el valor de la magnitud de gradiente es más pequeño que al menos uno de sus dos vecinos, en la dirección del ángulo obtenida en el paso anterior. De ser así, se asigna el valor 0 a dicho píxel, en caso contrario se asigna el valor que tenga la magnitud del gradiente.

La salida de este segundo paso es la imagen I_n con los bordes adelgazados, es decir, $\mathbf{E}_m(\mathbf{i},\mathbf{j})$, después de la supresión no máxima de puntos de borde.

Algoritmo: Supresión no máxima

Entrada: Imagen E_m de la magnitud del gradiente

Imagen E_0 de la orientación del gradiente

Salida: Imagen I_n

Considerar: Cuatro direcciones d_1, d_2, d_3, d_4 identificadas por las direcciones de $0^\circ, 45^\circ, 90^\circ$ y 135° , respecto al eje horizontal.

1. Para cada píxel (i, j) :

1.1 Encontrar la dirección d_k que mejor se aproxima a la dirección $E_0(i, j)$, que viene a ser la perpendicular del borde.

1.2 Si $E_m(i, j)$ es más pequeño que al menos uno de sus dos vecinos en la dirección d_k , al pixel (I, j) de I_n se le asigna el valor de 0, $I_n(i, j) = 0$ (supresión), de otro modo $I_n(i, j) = E_m(i, j)$.

2. Devolver I_n

Histéresis de umbral a la supresión no máxima

La imagen obtenida en el paso anterior, suele contener máximos locales creados por el ruido. Una solución para eliminar dicho ruido es la histéresis del umbral.

El proceso consiste en tomar la imagen obtenida del paso anterior, tomar la orientación de los puntos de borde de la imagen y tomar dos umbrales, el primero más pequeño que el segundo.

Para cada punto de la imagen, se debe localizar el siguiente punto de borde no explorado que sea mayor al segundo umbral. A partir de dicho punto, seguir las cadenas de máximos locales.

Algoritmo: Histéresis de umbral a la supresión no máxima

Entrada: imagen I_n obtenida del paso anterior

Imagen E_0 de la orientación del gradiente

Umbral t_1

Umbral t_2 , donde $t_1 < t_2$

Salida: imagen G con los bordes conectados de contornos

1. Para todos los puntos de I_n y explorando I_n en orden fijo:

1.1 Localizar el siguiente punto de borde no explorado previamente $I_n(i, j)$, tal que $I_n(i, j) > t_2$.

1.2 Comenzar a partir de $I_n(i, j)$, seguir las cadenas de los máximos locales conectados en ambas direcciones, perpendiculares a la normal del borde, siempre que $I_n > t_1$.

1.3 Marcar todos los puntos explorados y, salvar la lista de todos los puntos del entorno conectado encontrado.

2. Devolver G formada por el conjunto de bordes conectados de contornos de la imagen, así como la magnitud y orientación, describiendo las propiedades de los puntos de borde [7].

En la Fig. 1.24, se muestran, en el inciso **a** la imagen original, en el inciso **b** la orientación, en el inciso **c** la supresión no máxima, y el inciso **d** la histéresis de umbral, todas aplicadas a la imagen original.

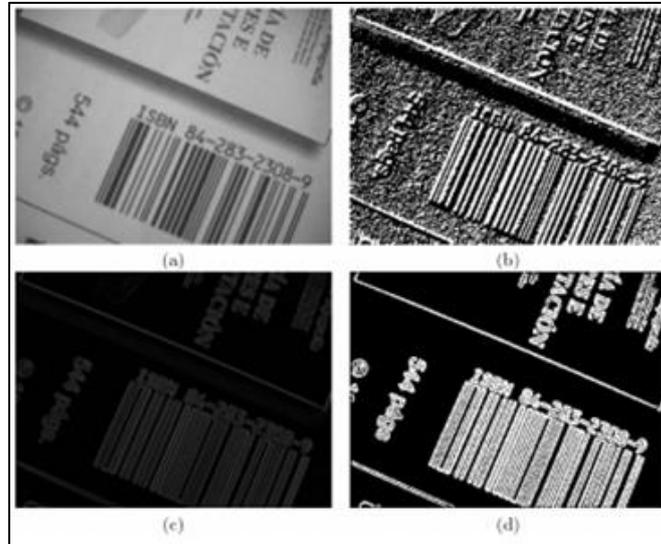


Fig. 1.24 Resultados con el algoritmo de Canny

Un cuarto paso

Frecuentemente, es común que un cuarto y último paso se realice en el algoritmo de Canny, este paso consiste en cerrar los contornos que pudiesen haber quedado abiertos por problemas de ruido.

Un método muy utilizado es el algoritmo de Deriche y Cocquerez. Este algoritmo utiliza como entrada, una imagen binarizada de contornos de un píxel de ancho. El algoritmo busca los extremos de los contornos abiertos y sigue la dirección del máximo gradiente, hasta cerrarlos con otro extremo abierto.

El procedimiento consiste en buscar para cada píxel uno de los ocho patrones posibles, que delimitan la continuación del contorno en tres direcciones posibles. Esto se logra con la convolución de cada píxel con una máscara específica que se muestra en la Fig. 1.25.

Cuando alguno de los tres puntos es ya un píxel de borde, se entiende que el borde se ha cerrado, de lo contrario, se elige el píxel con el valor máximo de gradiente y se marca como nuevo píxel de borde, y se aplica nuevamente la convolución. Estos pasos se repiten para todo extremo abierto, hasta encontrar su cierre o hasta llegar a cierto número de iteraciones determinado [7].

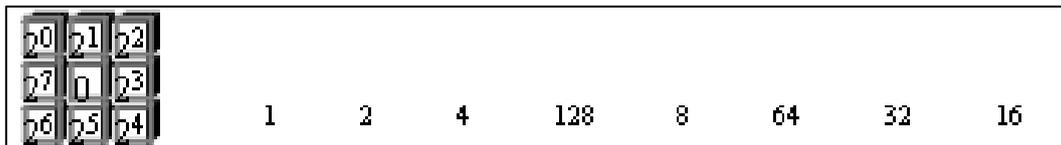


Fig. 1.25 Máscara de convolución para cierre de contornos

La Fig. 1.26, muestra el resultado final, de aplicar el operador de Canny a una imagen para detección de bordes.

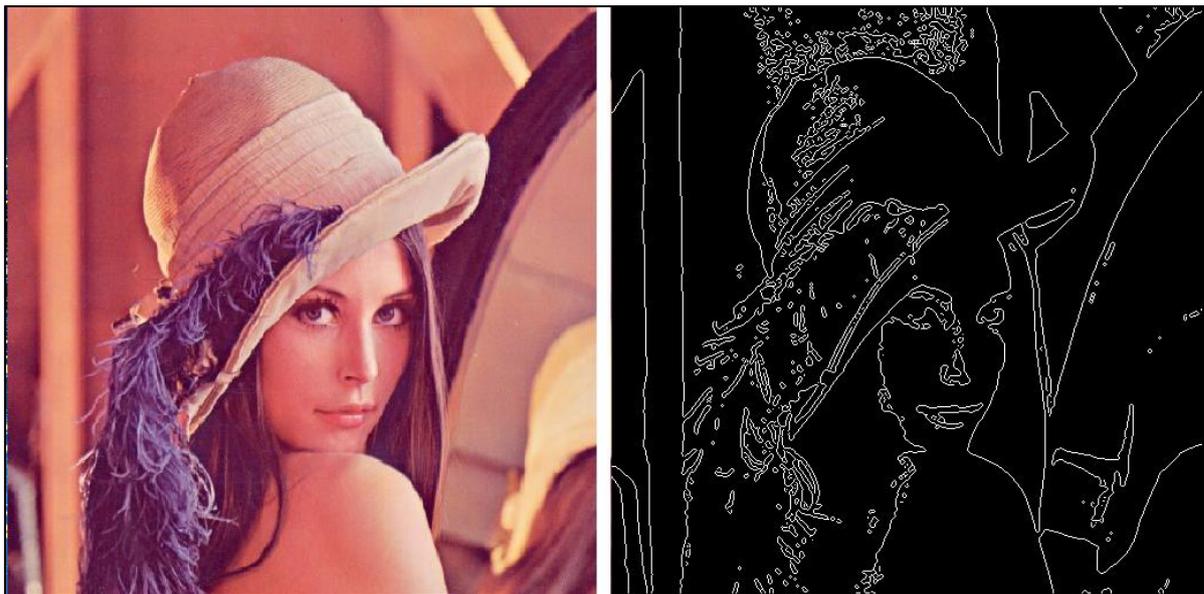


Fig. 1.26 Resultado aplicando el operador de Canny

1.2 Estado del Arte

En esta sección se presentan algunos avances en la detección de bordes en general, así como la importancia del estudio de éstos en general.

La detección de bordes, es un proceso fundamental en la industria de la metrología, ya que define las fronteras de la característica a ser medida. La precisión en la detección del borde, mejora la precisión en los procesos en curso, y los procedimientos de control de calidad.

En la actualidad, la detección de bordes ha sido implementada, mediante una gran variedad de métodos con los algoritmos de Canny, así como también el uso del Laplaciano.

✓ **Algoritmo para detección de bordes y ulterior determinación de objetos en imágenes digitales**

El algoritmo desarrollado inspecciona una imagen píxel a píxel, examinando aquellos que no hayan sido asignados a ningún objeto. Se realiza un recorrido secuencial de la matriz imagen y, para cada píxel con valor 1, se agregan sus coordenadas a una lista. Usando el concepto de 8-vecindad, se inspeccionan los vecinos buscando aquellos con valor 1 para agregarlos a la lista, al terminar de verificar los píxeles conectados al originalmente encontrado, se han almacenado las coordenadas de cada uno de los píxeles que forman un objeto. De esta manera, cada elemento de la lista es un objeto.

El programa desarrollado resulta robusto, demostrando alta eficiencia en distintas aplicaciones, siendo el tiempo computacional directamente proporcional al tamaño de los objetos, y con bajo consumo de memoria. Actualmente se trabaja en la relación área perímetro, para evitar contar como único dos objetos solapados [8].

✓ **Algoritmo de bordes activos (active contours - snakes)**

Se trata de un algoritmo iterativo de segmentación de imágenes, que toma como referencia una línea dada, para buscar el borde de mayor energía que se ajuste a dicha línea (búsqueda del borde más cercano a la línea) [9].

✓ ***Detección de placas de matrícula y su posterior emborronado***

Este trabajo aborda el problema de la detección de matrículas de placas. Para llevar a cabo tal tarea, desarrollaron una aplicación que a través de una serie de procesos aplicados de forma consecutiva, nos devuelve la imagen original con la zona de la placa de matrícula emborronada. Los procesos serán todos de carácter matemático o lógico. Hacen uso de un filtro Sobel para la identificación de bordes verticales, se elimina el ruido de la imagen, y a las zonas resultantes se aplica un proceso de dilatación, para adecuarlas a la forma de las matrículas de placas.

Tras esto, se identifican las distintas componentes conexas, de las que una es elegida como la placa de matrícula. Finalmente la componente elegida es emborronada. Resultados experimentales muestran que el método es robusto y eficiente [10].

✓ ***Un nuevo algoritmo de detección de bordes en imágenes con ruido gaussiano***

Los operadores de detección de bordes, como Roberts, Sobel, Prewit y LoG, tienen el problema de ser muy sensibles al ruido WG. El método que se propone en este artículo pretende solventar estas deficiencias, asegurando la continuidad, integridad y localización de los bordes de la imagen.

Muestra una buena comparación del método con los algoritmos clásicos de detección de bordes [11].

✓ ***Método de marcas de agua dual***

Se propone un algoritmo eficiente para detectar y recuperar imágenes con marcas de agua. En dicho algoritmo, cada bloque de la imagen contiene marca de agua de otros dos bloques, es decir, hay dos copias de la marca de agua por cada dos bloques no superpuestos. El método puede ser estructurado en tres pasos [12]:

1. Inclusión de la marca de agua.
2. Detección de los bloques manipulados.
3. Recuperación de los bloques manipulados.

✓ **Reconocimiento de imágenes mediante redes neuronales**

En este trabajo se pretende el reconocimiento de imágenes, mediante el algoritmo de Redes Neuronales. Para este trabajo se deberá llevar a cabo un estudio, del funcionamiento de las Redes Neuronales y configuración de ésta, para el correcto funcionamiento. Con lo que se pretende el desarrollo de un algoritmo, que clasifique imágenes mediante un aprendizaje previo. Para este aprendizaje se usará un conjunto de entrenamiento, con posibilidad de guardar el estado de la Red, una vez entrenada [13].

✓ ***Un método para eliminar ruido impulsivo en imágenes a color usando técnicas fuzzy***

Este método determina en primer lugar, un conjunto de píxeles libres de ruido, aplicando una condición restrictiva basada en el concepto de peer group, el cual es refinado posteriormente. Finalmente, los píxeles ruido son filtrados, maximizando el criterio de distancia fuzzy empleado [14].

✓ ***Algoritmo para reconocimiento de patrones y búsqueda de imágenes***

La búsqueda de imágenes o información a partir de otras imágenes, actualmente es una disciplina en pleno auge, grandes empresas tales como Google, llevan años investigando acerca de ello. Este artículo se centra en una imagen dada, se busca qué conjunto de imágenes la contienen, esto podría servir para encontrar objetos o personas dentro de paisajes, en videos de seguridad, etc... El método propuesto está basado en el algoritmo KRA, con el que se extrae la información que se quiere procesar. Una vez extraídas las características principales, éstas son invariantes respecto a la traslación, rotación y escalado. Para validar el método, se usa una imagen de referencia que será la que se buscará en una biblioteca de imágenes, en las que puede aparecer rotada, trasladada, escalada o, incluso, no aparecer.

El algoritmo indicará en cuáles de las imágenes de la biblioteca aparecen, con algún porcentaje, de confianza asociado [15].

Capítulo 2 Marco de Referencia

2.1-Conceptos básicos

El propósito fundamental de una memoria asociativa, es recuperar correctamente patrones completos, a partir de patrones de entrada, los cuales pueden estar alterados con ruido aditivo, sustractivo o combinado.

Una **Memoria asociativa** puede formularse como un sistema de entrada y salida, idea que se esquematiza en la Fig. 2.1.

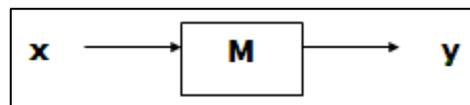


Fig. 2.1 Memoria asociativa

En este esquema, los patrones de entrada y salida están representados por vectores columna, denotados por \mathbf{x} y \mathbf{y} , respectivamente. Cada uno de los patrones de entrada, forma una asociación con el correspondiente patrón de salida, la cual es similar a la una pareja ordenada; por ejemplo, los patrones \mathbf{x} y \mathbf{y} del esquema anterior, forman la asociación (\mathbf{x}, \mathbf{y}) .

No obstante, que a lo largo de las dos secciones restantes del presente capítulo, se respetarán las notaciones originales de los autores de los modelos presentados aquí, a continuación se propone una notación, que se usará en la descripción de los conceptos básicos sobre memorias asociativas, y en el resto de los capítulos de esta tesis.

Los patrones de entrada y salida se denotarán con las letras negrillas, \mathbf{x} y \mathbf{y} , agregándoles números naturales como superíndices, para efectos de discriminación simbólica. Por ejemplo, a un patrón de entrada \mathbf{x}^1 le corresponderá el patrón de salida \mathbf{y}^1 , y ambos formarán la asociación $(\mathbf{x}^1, \mathbf{y}^1)$; del mismo modo, para un número entero positivo k específico, la asociación correspondiente será $(\mathbf{x}^k, \mathbf{y}^k)$.

La memoria asociativa \mathbf{M} se representa mediante una matriz, la cual se genera a partir de un conjunto finito de asociaciones (\mathbf{x}, \mathbf{y}) , conocidas de antemano: este es el **conjunto fundamental de aprendizaje**, o simplemente **conjunto fundamental**.

El conjunto fundamental se representa de la siguiente manera:

$$\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu = 1, 2, \dots, p\}$$

Donde p , es un número entero positivo que representa la cardinalidad del conjunto fundamental.

A los patrones que conforman las asociaciones del conjunto fundamental, se les llama **patrones fundamentales**. La naturaleza del conjunto fundamental, proporciona un importante criterio para clasificar las memorias asociativas:

Una memoria es **Autoasociativa**, si se cumple que $\mathbf{x}^\mu = \mathbf{y}^\mu \forall \mu \in \{1, 2, \dots, p\}$, por lo que uno de los requisitos que se debe de cumplir, es que $n = m$.

Una memoria **Heteroasociativa**, es aquella en donde $\exists \mu \in \{1, 2, \dots, p\}$ para el que se cumple que $\mathbf{x}^\mu \neq \mathbf{y}^\mu$. Nótese que puede haber memorias heteroasociativas con $n = m$.

En los problemas donde intervienen las memorias asociativas, se consideran dos fases importantes: La fase de aprendizaje, que es donde se genera la memoria asociativa a partir de las p asociaciones del conjunto fundamental, y la fase de recuperación, que es donde la memoria asociativa opera sobre un patrón de entrada, a la manera del esquema que aparece al inicio de esta sección.

A fin de especificar las componentes de los patrones, se requiere la notación para dos conjuntos a los que llamaremos arbitrariamente A y B . Las componentes de los vectores columna que representan a los patrones, tanto de entrada como de salida, serán elementos del conjunto A , y las entradas de la matriz M serán elementos del conjunto B .

No hay requisitos previos ni limitaciones respecto de la elección de estos dos conjuntos, por lo que no necesariamente deben ser diferentes o poseer características especiales. Esto significa que el número de posibilidades para escoger A y B es infinito.

Por convención, cada vector columna que representa a un patrón de entrada tendrá n componentes, cuyos valores pertenecen al conjunto A , y cada vector columna que representa a un patrón de salida, tendrá m componentes, cuyos valores pertenecen también al conjunto A . Es decir:

$$\mathbf{x}^\mu \in A^n \text{ y } \mathbf{y}^\mu \in A^m \forall \mu \in \{1, 2, \dots, p\}$$

La j -ésima componente de un vector columna, se indicará con la misma letra del vector, pero sin negrilla, colocando a j como subíndice ($j \in \{1, 2, \dots, n\}$ o $j \in \{1, 2, \dots, m\}$ según corresponda). La j -ésima componente del vector columna \mathbf{x} , se representa por: x_j^μ

Con los conceptos básicos ya descritos y con la notación anterior, es posible expresar las dos fases de una memoria asociativa:

1. **Fase de aprendizaje** (Generación de la memoria asociativa). Encontrar los operadores adecuados y una manera de generar una matriz \mathbf{M} , que almacene las p asociaciones del conjunto fundamental $\{(\mathbf{x}^1, \mathbf{y}^1), (\mathbf{x}^2, \mathbf{y}^2), \dots, (\mathbf{x}^p, \mathbf{y}^p)\}$, donde $\mathbf{x}^\mu \in A^n$ y $\mathbf{y}^\mu \in A^m \forall \mu \in \{1, 2, \dots, p\}$. Si $\exists \mu \in \{1, 2, \dots, p\}$, tal que $\mathbf{x}^\mu \neq \mathbf{y}^\mu$, la memoria será heteroasociativa, si $m = n$, y $\mathbf{x}^\mu = \mathbf{y}^\mu \forall \mu \in \{1, 2, \dots, p\}$, la memoria será autoasociativa.
2. **Fase de recuperación** (Operación de la memoria asociativa). Hallar los operadores adecuados y las condiciones suficientes, para obtener el patrón fundamental de salida \mathbf{y}^μ , cuando se opera la memoria \mathbf{M} con el patrón fundamental de entrada \mathbf{x}^μ ; lo anterior, para todos los elementos del conjunto fundamental y para ambos modos: autoasociativo y heteroasociativo.

Se dice que una memoria asociativa \mathbf{M} exhibe **recuperación correcta**, si al presentarle como entrada, en la fase de recuperación, un patrón \mathbf{x}^ω con $\omega \in \{1, 2, \dots, p\}$, ésta responde con el correspondiente patrón fundamental de salida \mathbf{y}^ω .

Una memoria asociativa bidireccional también es un sistema de entrada y salida, solamente que el proceso es bidireccional. La dirección hacia adelante, se describe de la misma forma que una memoria asociativa común: al presentarle una entrada \mathbf{x} , el sistema entrega una salida \mathbf{y} . La dirección hacia atrás, se lleva a cabo presentándole al sistema una entrada \mathbf{y} , para recibir una salida \mathbf{x} .

2.2 Memorias asociativas

A continuación, en esta sección haremos un breve recorrido por los modelos de memorias asociativas, con objeto de establecer el marco de referencia, en el que surgieron las memorias asociativas.

Las memorias asociativas que se presentarán en esta sección, son los modelos más representativos, que sirvieron de base para la creación de modelos matemáticos, que sustentan el diseño y operación de memorias asociativas más complejas. Para cada modelo, se describe su fase de aprendizaje y su fase de recuperación.

Se incluyen 4 modelos clásicos, basados en el anillo de los números racionales con las operaciones de multiplicación y adición: *Lernmatrix*, *Correlograph*, *Linear Associator* y Memoria Hopfield, además de tres modelos basados en paradigmas diferentes a la suma de productos: memorias asociativas Morfológicas, memorias asociativas Alfa-Beta y memorias asociativas Media.

2.2.1 *Lernmatrix* de Steinbuch

Karl Steinbuch, fue uno de los primeros investigadores en desarrollar un método, para codificar información en arreglos cuadrículados conocidos como *crossbar*. La importancia de la *Lernmatrix*, se evidencia en una afirmación que hace Kohonen en su artículo de 1972, donde apunta que las matrices de correlación, base fundamental de su innovador trabajo, vinieron a sustituir a la *Lernmatrix* de Steinbuch.

La *Lernmatrix* es una memoria heteroasociativa, que puede funcionar como un clasificador de patrones binarios, si se escogen adecuadamente los patrones de salida; es un sistema de entrada y salida que al operar, acepta como entrada un patrón binario $\mathbf{x}^\mu \in A^n$, $A = \{0,1\}$ y produce como salida la clase $\mathbf{y}^\mu \in A^p$ que le corresponde (de entre p clases diferentes), codificada ésta, con un método que en la literatura se le ha llamado *one-hot*. El método funciona así: para representar la clase $k \in \{1, 2, \dots, p\}$, se asignan a las componentes del vector de salida \mathbf{y}^μ , los siguientes valores: $y_k^\mu = 1$, y $y_j^\mu = 0$ para $j = 1, 2, \dots, k-1, k+1, \dots, p$.

Algoritmo de la *Lernmatrix*

Fase de aprendizaje

Se genera el esquema (*crossbar*), al incorporar la pareja de patrones de entrenamiento $(\mathbf{x}^\mu, \mathbf{y}^\mu) \in A^n \times A^p$. Cada uno de los componentes m_{ij} de \mathbf{M} , la *Lernmatrix* de Steinbuch, tiene valor cero al inicio, y se actualiza de acuerdo con la regla $m_{ij} + \Delta m_{ij}$, descrita en la siguiente expresión, donde:

$$\Delta m_{ij} = \begin{cases} +\varepsilon & \text{si } x_j^\mu = 1 = y_i^\mu \\ -\varepsilon & \text{si } x_j^\mu = 0 \text{ y } y_i^\mu = 1 \\ 0 & \text{en otro caso} \end{cases}$$

Donde ε es una constante positiva escogida previamente: es usual que ε es igual a 1.

Fase de recuperación

La i -ésima coordenada y_i^ω del vector de clase $\mathbf{y}^\omega \in A^p$, se obtiene como lo indica la siguiente expresión, donde \vee es el operador *máximo*:

$$y_i^\omega = \begin{cases} 1 & \text{si } \sum_{j=1}^n m_{ij} \cdot x_j^\omega = \vee_{h=1}^p \left[\sum_{j=1}^n m_{hj} \cdot x_j^\omega \right] \\ 0 & \text{en otro caso} \end{cases}$$

2.2.2 *Correlograph* de Willshaw, Buneman y Longuet-Higgins

El *correlograph* es un dispositivo óptico elemental, capaz de funcionar como una memoria asociativa. En palabras de los autores; "el sistema es tan simple, que podría ser construido en cualquier laboratorio escolar de física elemental".

Algoritmo del *Correlograph*

Fase de aprendizaje

La *red asociativa*, se genera al incorporar la pareja de patrones de entrenamiento $(\mathbf{x}^\mu, \mathbf{y}^\mu) \in A^n \times A^m$. Cada uno de los componentes m_{ij} de la *red asociativa* \mathbf{M} tiene valor cero al inicio, y se actualiza de acuerdo con la regla descrita en la expresión:

$$m_{ij} = \begin{cases} 1 & \text{si } y_i^\mu = 1 = x_j^\mu \\ \text{valor anterior} & \text{en otro caso} \end{cases}$$

Fase de recuperación

Se le presenta a la *red asociativa* \mathbf{M} un vector de entrada $\mathbf{x}^o \in A^n$. Se realiza el producto de la matriz \mathbf{M} por el vector \mathbf{x}^o , y se ejecuta una operación de umbralizado, de acuerdo con la siguiente expresión:

$$y_i^o = \begin{cases} 1 & \text{si } \sum_{j=1}^n m_{ij} \cdot x_j^o \geq u \\ 0 & \text{en otro caso} \end{cases}$$

Donde u es el valor de umbral. Una estimación aproximada del valor de umbral u , se puede lograr con la ayuda de un número, mencionado en el artículo de Willshaw *et al.* de 1969: $\log_2 n$

2.2.3 Linear Associator de Anderson-Kohonen

El *Linear Associator* tiene su origen en los trabajos pioneros de 1972, publicados por Anderson y Kohonen.

Para presentar el *Linear Associator*, consideremos de nuevo el conjunto fundamental:

$$\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu = 1, 2, \dots, p\} \text{ con } A = \{0, 1\}, \mathbf{x}^\mu \in A^n \text{ y } \mathbf{y}^\mu \in A^m$$

Algoritmo del Linear Associator

Fase de aprendizaje

- 1) Para cada una de las p asociaciones $(\mathbf{x}^\mu, \mathbf{y}^\mu)$, se encuentra la matriz $\mathbf{y}^\mu \cdot (\mathbf{x}^\mu)^T$ de dimensiones $m \times n$
- 2) Se suman las p matrices para obtener la memoria \mathbf{M} , mediante la ecuación 19.

$$\mathbf{M} = \sum_{\mu=1}^p \mathbf{y}^\mu \cdot (\mathbf{x}^\mu)^T = [m_{ij}]_{m \times n} \quad (16)$$

De manera que la ij -ésima componente de la memoria \mathbf{M} , se expresa así mediante la siguiente ecuación 17:

$$m_{ij} = \sum_{\mu=1}^p y_i^\mu x_j^\mu \quad (17)$$

Fase de recuperación

Esta fase consiste en presentarle a la memoria un patrón de entrada \mathbf{x}^ω , donde $\omega \in \{1, 2, \dots, p\}$ y realizar la operación descrita en la ecuación 18.

$$\mathbf{M} \cdot \mathbf{x}^\omega = \left[\sum_{\mu=1}^p \mathbf{y}^\mu \cdot (\mathbf{x}^\mu)^t \right] \cdot \mathbf{x}^\omega \quad (18)$$

2.2.4 La memoria asociativa Hopfield

El artículo de John J. Hopfield de 1982, publicado por la prestigiosa y respetada *National Academy of Sciences* (en sus *Proceedings*), impactó positivamente, y trajo a la palestra internacional su famosa memoria asociativa.

En el modelo que originalmente propuso Hopfield, cada neurona x_i tiene dos posibles estados, a la manera de las neuronas de McCulloch-Pitts: $x_i = 0$ y $x_i = 1$; sin embargo, Hopfield observa que, para un nivel dado de exactitud en la recuperación de patrones, la capacidad de almacenamiento de información de la memoria, se puede incrementar por un factor de 2, si se escogen como posibles estados de las neuronas los valores $x_i = -1$ y $x_i = 1$, en lugar de los valores originales $x_i = 0$ y $x_i = 1$.

Al utilizar el conjunto $\{-1, 1\}$ y el valor de umbral cero, la fase de aprendizaje para la memoria Hopfield, será similar, en cierta forma, a la fase de aprendizaje del *Linear Associator*. La intensidad de la fuerza de conexión de la neurona x_i a la neurona x_j , se representa por el valor de m_{ij} , y se considera que hay simetría, es decir, $m_{ij} = m_{ji}$.

Si x_i no está conectada con x_j , entonces $m_{ij} = 0$; en particular, no hay conexiones recurrentes de una neurona a sí misma, lo cual significa que $m_{ij} = 0$. El estado instantáneo del sistema, está completamente especificado por el vector columna de dimensión n , cuyas coordenadas son los valores de las n neuronas.

La memoria Hopfield es autoasociativa, simétrica, con ceros en la diagonal principal. En virtud de que la memoria es autoasociativa, el conjunto fundamental para la memoria Hopfield, es $\{(\mathbf{x}^\mu, \mathbf{x}^\mu) \mid \mu = 1, 2, \dots, p\}$ con $\mathbf{x}^\mu \in A^n$ y $A = \{-1, 1\}$

Algoritmo Hopfield

Fase de aprendizaje

La fase de aprendizaje para la memoria Hopfield, es similar a la fase de aprendizaje del *Linear Associator*, con una ligera diferencia relacionada con la diagonal principal en ceros, como se muestra en la siguiente regla, para obtener la ij -ésima componente de la memoria Hopfield \mathbf{M} :

$$m_{ij} = \begin{cases} \sum_{\mu=1}^p x_i^{\mu} x_j^{\mu} & \text{si } i \neq j \\ 0 & \text{si } i = j \end{cases}$$

Fase de recuperación

Si se le presenta un patrón de entrada \mathbf{x} a la memoria Hopfield, ésta cambiará su estado con el tiempo, de modo que cada neurona x_i ajuste su valor, de acuerdo con el resultado que arroje la comparación de la cantidad $\sum_{j=1}^n m_{ij} x_j$ con un valor de umbral, el cual normalmente se coloca en cero.

Se representa el estado de la memoria Hopfield en el tiempo t por $\mathbf{x}(t)$; entonces $x_i(t)$, representa el valor de la neurona x_i en el tiempo t , y $x_i(t+1)$, el valor de x_i en el tiempo siguiente ($t+1$).

Dado un vector columna de entrada \mathbf{x} , la fase de recuperación consta de tres pasos:

- 1) Para $t = 0$, se hace $\mathbf{x}(t) = \mathbf{x}$; es decir, $x_i(0) = \tilde{x}_i, \forall i \in \{1, 2, 3, \dots, n\}$
- 2) $\forall i \in \{1, 2, 3, \dots, n\}$, se calcula $x_i(t+1)$ de acuerdo con la condición siguiente:

$$x_i(t+1) = \begin{cases} 1 & \text{si } \sum_{j=1}^n m_{ij} x_j(t) > 0 \\ x_i(t) & \text{si } \sum_{j=1}^n m_{ij} x_j(t) = 0 \\ -1 & \text{si } \sum_{j=1}^n m_{ij} x_j(t) < 0 \end{cases}$$

- 3) Se compara $x_i(t+1)$ con $x_i(t) \forall i \in \{1, 2, 3, \dots, n\}$. Si $\mathbf{x}(t+1) = \mathbf{x}(t)$, el proceso termina y el vector recuperado es $\mathbf{x}(0) = \mathbf{x}$. De otro modo, el proceso continúa de la siguiente manera: los pasos 2 y 3 se iteran tantas veces como sea necesario, hasta llegar a un valor $t = \tau$, para el cual $x_i(\tau+1) = x_i(\tau) \forall i \in \{1, 2, 3, \dots, n\}$; el proceso termina y el patrón recuperado es $\mathbf{x}(\tau)$.

En el artículo original de 1982, Hopfield había estimado, que su memoria tenía una capacidad de recuperar $0.15n$ patrones, y en el trabajo de Abu-Mostafa & St. Jacques, se estableció formalmente que una cota superior, para el número de vectores de estado arbitrarios estables en una memoria Hopfield, es n .

2.2.5 Memorias asociativas Morfológicas

La diferencia fundamental entre las memorias asociativas clásicas (*Lernmatrix*, *Correlograph*, *Linear Associator* y Memoria Asociativa Hopfield) y las memorias asociativas morfológicas, radica en los fundamentos operacionales de éstas últimas, que son las operaciones morfológicas de *dilatación* y *erosión*; el nombre de las memorias asociativas morfológicas, está inspirado precisamente en estas dos operaciones básicas. Estas memorias rompieron con el esquema utilizado a través de los años, en los modelos de memorias asociativas clásicas, que utilizan operaciones convencionales entre vectores y matrices, para la fase de aprendizaje y suma de productos, para la recuperación de patrones. Las memorias asociativas morfológicas, cambian los productos por sumas y las sumas por máximos o mínimos en ambas fases, tanto de aprendizaje como de recuperación.

Hay dos tipos de memorias asociativas morfológicas: las memorias *max*, simbolizadas con **M**, y las memorias *min*, cuyo símbolo es **W**; en cada uno de los dos tipos, las memorias pueden funcionar en ambos modos: heteroasociativo y autoasociativo.

Se definen dos nuevos productos matriciales:

El *producto máximo* entre **D** y **H**, denotado por **C** = **D** ∇ **H**, es una matriz $[c_{ij}]_{m \times n}$ cuya *ij*-ésima componente c_{ij} , es dada por la ecuación 19:

$$c_{ij} = \bigvee_{k=1}^r (d_{ik} + h_{kj}) \quad (19)$$

El *producto mínimo* de \mathbf{D} y \mathbf{H} denotado por $\mathbf{C} = \mathbf{D} \Delta \mathbf{H}$, es una matriz $[C_{ij}]_{m \times n}$ cuya ij -ésima componente c_{ij} , es dada por la ecuación 20:

$$c_{ij} = \bigwedge_{k=1}^r (d_{ik} + h_{kj}) \quad (20)$$

Los productos máximo y mínimo contienen a los operadores máximo y mínimo, los cuales están íntimamente ligados, con los conceptos de las dos operaciones básicas de la morfología matemática: *dilatación* y *erosión*, respectivamente.

2.2.5.1 Memorias heteroasociativas Morfológicas

Algoritmo de las memorias Morfológicas Max

Fase de aprendizaje

1. Para cada una de las p asociaciones $(\mathbf{x}^\mu, \mathbf{y}^\mu)$, se usa el producto mínimo para crear la matriz $\mathbf{y}^\mu \Delta (-\mathbf{x}^\mu)^t$ de dimensiones $m \times n$, donde el negado transpuesto del patrón de entrada \mathbf{x}^μ , se define como:

$$(-\mathbf{x}^\mu)^t = (-x_1^\mu, -x_2^\mu, \dots, x_n^\mu)$$

2. Se aplica el operador máximo \vee a las p matrices para obtener la memoria \mathbf{M} , como se muestra en la ecuación 21.

$$\mathbf{M} = \bigvee_{\mu=1}^p [\mathbf{y}^\mu \Delta (-\mathbf{x}^\mu)^t] \quad (21)$$

Fase de Recuperación

Esta fase consiste, en realizar el producto mínimo Δ de la memoria \mathbf{M} con el patrón de entrada \mathbf{x}^ω , donde $\omega \in \{1, 2, \dots, p\}$, para obtener un vector columna \mathbf{y} de dimensión m , por medio de la ecuación 22:

$$\mathbf{y} = \mathbf{M} \Delta \mathbf{x}^\omega \quad (22)$$

Las fases de aprendizaje y de recuperación de las **memorias Morfológicas min**, se obtienen por dualidad.

2.2.5.2 Memorias auto-asociativas Morfológicas

Para este tipo de memorias se utilizan los mismos algoritmos descritos anteriormente, y que son aplicados a las memorias heteroasociativas; lo único que cambia es el conjunto fundamental. Para este caso, se considera el siguiente conjunto fundamental:

$$\{(\mathbf{x}^\mu, \mathbf{x}^\mu) \mid \mathbf{x}^\mu \in A^n, \text{ donde } \mu = 1, 2, \dots, p\}$$

2.2.6 Memorias asociativas Alfa-Beta

Las memorias Alfa-Beta utilizan máximos y mínimos, y dos operaciones binarias originales α y β , de las cuales heredan el nombre.

Para la definición de las operaciones binarias α y β , se deben especificar los conjuntos A y B , los cuales son:

$$A = \{0, 1\} \quad \text{y} \quad B = \{0, 1, 2\}$$

La operación binaria $\alpha: A \times A \rightarrow B$ se define en la Fig. 2.2, como:

x	y	$\alpha(x, y)$
0	0	1
0	1	0
1	0	2
1	1	1

Fig. 2.2 Fase de aprendizaje Alfa-Beta

La operación binaria $\beta: B \times A \rightarrow A$ se define en la Fig. 2.3, como:

X	y	$\beta(x, y)$
0	0	0
0	1	0
1	0	0
1	1	1
2	0	1
2	1	1

Fig. 2.3 Fase de recuperación Alfa-Beta

Los conjuntos A y B , las operaciones binarias α y β junto con los operadores \wedge (mínimo) y \vee (máximo) usuales, conforman el sistema algebraico $(A, B, \alpha, \beta, \wedge, \vee)$, en el que están inmersas las memorias asociativas Alfa-Beta.

Ejemplo

Dados los siguientes patrones, aplicar el algoritmo de memorias asociativas Alfa-Beta.

$$x_1 = \begin{array}{|c|} \hline 0 \\ \hline 1 \\ \hline 0 \\ \hline 1 \\ \hline \end{array} \quad \rightarrow \quad y_1 = \begin{array}{|c|} \hline 1 \\ \hline 0 \\ \hline 0 \\ \hline \end{array}$$

$$x_2 = \begin{array}{|c|} \hline 0 \\ \hline 1 \\ \hline 1 \\ \hline 1 \\ \hline \end{array} \quad \rightarrow \quad y_2 = \begin{array}{|c|} \hline 1 \\ \hline 1 \\ \hline 0 \\ \hline \end{array}$$

$$x_3 = \begin{array}{|c|} \hline 1 \\ \hline 0 \\ \hline 0 \\ \hline 1 \\ \hline \end{array} \quad \rightarrow \quad y_3 = \begin{array}{|c|} \hline 0 \\ \hline 0 \\ \hline 1 \\ \hline \end{array}$$

Asociamos cada patrón de salida, con su entrada transpuesta, como se muestra a continuación.

$$y_1 \alpha (x_1)^t = \begin{array}{|c|} \hline 1 \\ \hline 0 \\ \hline 0 \\ \hline \end{array} \quad \alpha \quad \begin{array}{|c|c|c|c|} \hline 0 & 1 & 0 & 1 \\ \hline \end{array} \quad = \quad \begin{array}{|c|c|c|c|} \hline \alpha(1,0) & \alpha(1,1) & \alpha(1,0) & \alpha(1,1) \\ \hline \alpha(0,0) & \alpha(0,1) & \alpha(0,0) & \alpha(0,1) \\ \hline \alpha(0,0) & \alpha(0,1) & \alpha(0,0) & \alpha(0,1) \\ \hline \end{array}$$

$$y_2 \alpha (x_2)^t = \begin{array}{|c|} \hline 1 \\ \hline 1 \\ \hline 0 \\ \hline \end{array} \quad \alpha \quad \begin{array}{|c|c|c|c|} \hline 0 & 1 & 1 & 1 \\ \hline \end{array} \quad = \quad \begin{array}{|c|c|c|c|} \hline \alpha(1,0) & \alpha(1,1) & \alpha(1,1) & \alpha(1,1) \\ \hline \alpha(1,0) & \alpha(1,1) & \alpha(1,1) & \alpha(1,1) \\ \hline \alpha(0,0) & \alpha(0,1) & \alpha(0,1) & \alpha(0,1) \\ \hline \end{array}$$

$$y_3 \alpha (x_3)^t = \begin{array}{|c|} \hline 0 \\ \hline 0 \\ \hline 1 \\ \hline \end{array} \quad \alpha \quad \begin{array}{|c|c|c|c|} \hline 1 & 0 & 0 & 1 \\ \hline \end{array} \quad = \quad \begin{array}{|c|c|c|c|} \hline \alpha(0,1) & \alpha(0,0) & \alpha(0,0) & \alpha(0,1) \\ \hline \alpha(0,1) & \alpha(0,0) & \alpha(0,0) & \alpha(0,1) \\ \hline \alpha(1,1) & \alpha(1,0) & \alpha(1,0) & \alpha(1,1) \\ \hline \end{array}$$

Aplicando la fase de aprendizaje (Fig. 2.2) a las matrices resultantes, tenemos que:

$$y1\alpha(x1)^t = \begin{array}{|c|c|c|c|} \hline 2 & 1 & 2 & 1 \\ \hline 1 & 0 & 1 & 0 \\ \hline 1 & 0 & 1 & 0 \\ \hline \end{array}$$

$$y2\alpha(x2)^t = \begin{array}{|c|c|c|c|} \hline 2 & 1 & 1 & 1 \\ \hline 2 & 1 & 1 & 1 \\ \hline 1 & 0 & 0 & 0 \\ \hline \end{array}$$

$$y3\alpha(x3)^t = \begin{array}{|c|c|c|c|} \hline 0 & 1 & 1 & 0 \\ \hline 0 & 1 & 1 & 0 \\ \hline 1 & 2 & 2 & 1 \\ \hline \end{array}$$

Enseguida, aplicando la regla de máximos y mínimos a las 3 matrices resultantes, obtenemos dos matrices, una matriz correspondiente a máximos y la segunda correspondiente a mínimos, entonces tenemos:

$$M = \begin{array}{|c|c|c|c|} \hline 2 & 1 & 2 & 1 \\ \hline 1 & 0 & 1 & 0 \\ \hline 1 & 0 & 1 & 0 \\ \hline \end{array} \vee \begin{array}{|c|c|c|c|} \hline 2 & 1 & 1 & 1 \\ \hline 2 & 1 & 1 & 1 \\ \hline 1 & 0 & 0 & 0 \\ \hline \end{array} \vee \begin{array}{|c|c|c|c|} \hline 0 & 1 & 1 & 0 \\ \hline 0 & 1 & 1 & 0 \\ \hline 1 & 2 & 2 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 2 & 1 & 2 & 1 \\ \hline 2 & 1 & 1 & 1 \\ \hline 1 & 2 & 2 & 1 \\ \hline \end{array}$$

$$W = \begin{array}{|c|c|c|c|} \hline 2 & 1 & 2 & 1 \\ \hline 1 & 0 & 1 & 0 \\ \hline 1 & 0 & 1 & 0 \\ \hline \end{array} \wedge \begin{array}{|c|c|c|c|} \hline 2 & 1 & 1 & 1 \\ \hline 2 & 1 & 1 & 1 \\ \hline 1 & 0 & 0 & 0 \\ \hline \end{array} \wedge \begin{array}{|c|c|c|c|} \hline 0 & 1 & 1 & 0 \\ \hline 0 & 1 & 1 & 0 \\ \hline 1 & 2 & 2 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline 0 & 1 & 1 & 0 \\ \hline 0 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 0 \\ \hline \end{array}$$

Una vez terminado este paso, podemos pasar a la fase de recuperación (fig 2.3). Para poder llevar a cabo esta etapa, operamos nuestro patrón de entrada \mathbf{x} , con la matriz de máximos, o también lo podemos hacer con la matriz de mínimos.

Para recuperar con la matriz de máximos, empleamos el operador de mínimos, para obtener el resultado, y para recuperar, con la matriz de mínimos, empleamos el operador de máximos, para obtener el resultado, como se muestra a continuación:

Con máximos tenemos:

$$M \wedge \beta(x1) = \begin{array}{|c|c|c|c|} \hline 2 & 1 & 2 & 1 \\ \hline 2 & 1 & 1 & 1 \\ \hline 1 & 2 & 2 & 1 \\ \hline \end{array} \wedge \beta \begin{array}{|c|} \hline 0 \\ \hline 1 \\ \hline 0 \\ \hline 1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|c|c|c|} \hline \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(2,0) & \wedge & \beta(1,1) \\ \hline \beta(2,0) & \wedge & \beta(1,1) & \wedge & \beta(1,0) & \wedge & \beta(1,1) \\ \hline \beta(1,0) & \wedge & \beta(2,1) & \wedge & \beta(2,0) & \wedge & \beta(1,1) \\ \hline \end{array}$$

$$M \wedge \beta(x1) = \begin{array}{|c|c|c|c|c|c|c|} \hline 1 & \wedge & 1 & \wedge & 1 & \wedge & 1 \\ \hline 1 & \wedge & 1 & \wedge & 0 & \wedge & 1 \\ \hline 0 & \wedge & 1 & \wedge & 1 & \wedge & 1 \\ \hline \end{array} = \begin{array}{|c|} \hline 1 \\ \hline 0 \\ \hline 0 \\ \hline \end{array} = y1$$

Con mínimos tenemos:

$$W \vee \beta(x1) = \begin{array}{|c|c|c|c|} \hline 0 & 1 & 1 & 0 \\ \hline 0 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 0 \\ \hline \end{array} \vee \beta \begin{array}{|c|} \hline 0 \\ \hline 1 \\ \hline 0 \\ \hline 1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|c|c|} \hline \beta(0,0) & \vee & \beta(1,1) & \vee & \beta(1,0) & \vee & \beta(0,1) \\ \hline \beta(0,0) & \vee & \beta(0,1) & \vee & \beta(1,0) & \vee & \beta(0,1) \\ \hline \beta(1,0) & \vee & \beta(0,1) & \vee & \beta(0,0) & \vee & \beta(0,1) \\ \hline \end{array}$$

$$W \vee \beta(x1) = \begin{array}{|c|c|c|c|c|c|c|} \hline 0 & N & 1 & \vee & 0 & \vee & 0 \\ \hline 0 & N & 0 & \vee & 0 & \vee & 0 \\ \hline 0 & N & 0 & \vee & 0 & \vee & 0 \\ \hline \end{array} = \begin{array}{|c|} \hline 1 \\ \hline 0 \\ \hline 0 \\ \hline \end{array} = y1$$

Cuando la memoria es autoasociativa, quiere decir, que los patrones de entrada y salida serán los mismos, la única diferencia es que la entrada, será el vector transpuesto de la salida, por ejemplo el siguiente patrón:

$$x1 = \begin{array}{|c|} \hline 0 \\ \hline 1 \\ \hline 0 \\ \hline 1 \\ \hline \end{array}$$

La operación para cada patrón será de la siguiente manera:

$$x1 \alpha (x1)^t = \begin{array}{|c|} \hline 0 \\ \hline 1 \\ \hline 0 \\ \hline 1 \\ \hline \end{array} \alpha \begin{array}{|c|c|c|c|} \hline 0 & 1 & 0 & 1 \\ \hline \end{array}$$

De esta manera cada patrón será operado por sí mismo, y los pasos posteriores, serán los mismos, que los ya descritos anteriormente.

2.2.7 Memorias Asociativas Media

Las Memorias Asociativas Media utilizan los operadores A y B, definidos en la ecuación 23 de la siguiente forma:

$$\begin{array}{l} A(x, y) = x - y \\ B(x, y) = x + y \end{array} \quad (23)$$

Las operaciones utilizadas se describen a continuación.

Sean $P = [p_{ij}]_{m \times r}$ y $Q = [q_{ij}]_{r \times n}$ dos matrices.

Operación \diamond_A : $P_{m \times r} \diamond_A Q_{r \times n} = [f_{ij}^A]_{m \times n}$, donde $f_{ij}^A = \mathbf{med}_{k=1}^r A(p_{ik}, q_{k,j})$

Operación \diamond_B : $P_{m \times r} \diamond_B Q_{r \times n} = [f_{ij}^B]_{m \times n}$, donde $f_{ij}^B = \mathbf{med}_{k=1}^r B(p_{ik}, q_{k,j})$

Algoritmo memorias media

Fase de aprendizaje

Paso 1. Para cada $\xi = 1, 2, \dots, p$, de cada pareja $(\mathbf{x}^\xi, \mathbf{y}^\xi)$, se construye la matriz:

$$[\mathbf{y}^\xi \diamond_A (\mathbf{x}^\xi)^t]_{m \times n}$$

Paso 2. Se aplica el operador media a las matrices obtenidas en el paso 1, para obtener la matriz \mathbf{M} , mediante la ecuación 24:

$$\mathbf{M} = \mathbf{med}_{\xi=1}^p \left[\mathbf{y}^\xi \diamond_A (\mathbf{x}^\xi)^t \right] \quad (24)$$

El ij -ésimo componente \mathbf{M} está dado por la ecuación 25:

$$m_{ij} = \mathbf{med}_{\xi=1}^p A(y_i^\xi, x_j^\xi) \quad (25)$$

Fase de recuperación

Se tienen dos casos:

Caso 1. Recuperación de un patrón fundamental. Un patrón \mathbf{x}^w , con $w \in \{1, 2, \dots, p\}$ se le presenta a la memoria \mathbf{M} , y se realiza la siguiente operación:

$$\mathbf{M} \diamond_B \mathbf{x}^w$$

El resultado es un vector columna de dimensión n , con la i -ésima componente, dado por la ecuación 29 como:

$$\left(\mathbf{M} \diamond_B \mathbf{x}^w\right)_i = \mathbf{med}_B(m_{ij}, x_j^w) \quad (26)$$

Caso 2. Recuperación de un patrón alterado. Un patrón $\tilde{\mathbf{x}}$, que es una versión alterada de un patrón \mathbf{x}^w , se le presenta a la memoria \mathbf{M} y se realiza la siguiente operación:

$$\mathbf{M} \diamond_B \tilde{\mathbf{x}}$$

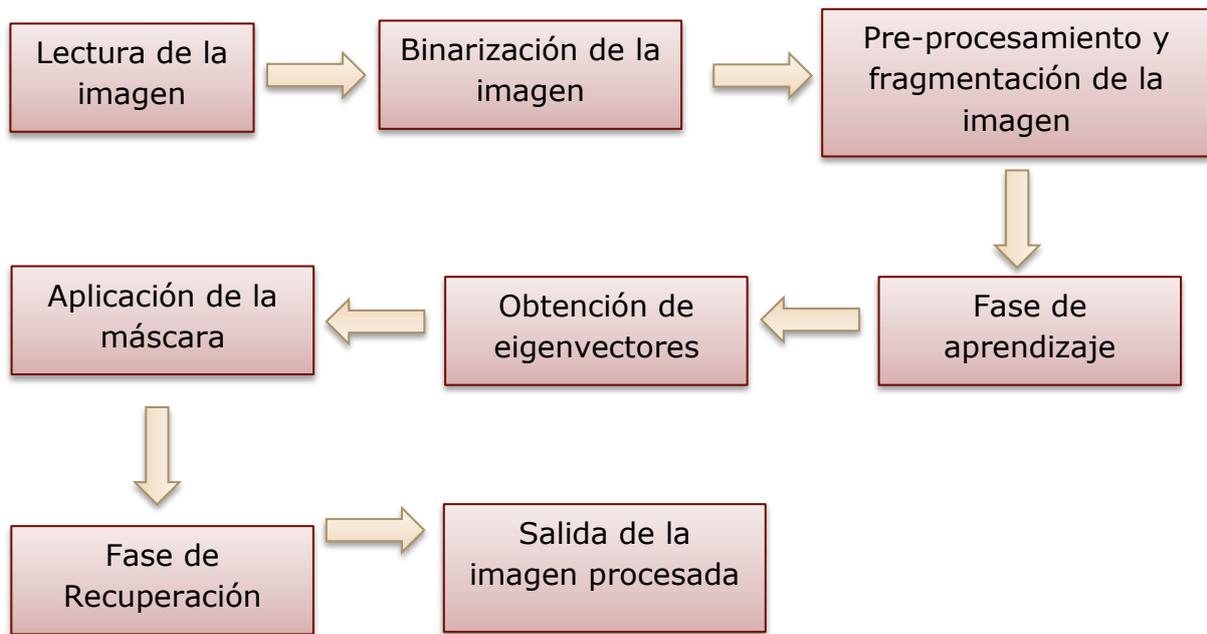
De nuevo, el resultado es un vector de dimensión n , con la i -ésima, dado por la ecuación 27, como:

$$\left(\mathbf{M} \diamond_B \tilde{\mathbf{x}}\right)_i = \mathbf{med}_B(m_{ij}, \tilde{x}_j) \quad (27)$$

Capítulo 3 Desarrollo

En este capítulo se describe el desarrollo del método abordado en este trabajo, que consiste en un algoritmo para la detección de bordes, mediante el uso de memorias asociativas Alfa-Beta.

Una breve explicación del desarrollo se presenta en el siguiente diagrama a bloques.



Para el desarrollo de este algoritmo se utilizó MATLAB. Como en esta aplicación se trabajó con imágenes, esta herramienta es adecuada, ya que se pueden manejar matrices, de una manera sencilla y práctica, ya que al trabajar con matrices de n dimensiones, facilita en gran medida cálculos, que resultan ser laboriosos entre más grande es la imagen, para el trabajo se utilizó la plataforma Windows.

Se utiliza esta herramienta por que entre sus prestaciones básicas se hallan: la manipulación de matrices(lo cual fue de gran ayuda), la representación de datos, funciones, la implementación de algoritmos y la creación de interfaces de usuario (GUI). El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI). Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas (toolboxes); y las de Simulink con los paquetes de bloques (blocksets).

MATLAB es un software muy usado en universidades, y centros de investigación y desarrollo. En los últimos años ha aumentado el número de prestaciones, como la de programar directamente procesadores digitales de señal.

3.1 Obtención, conversión y segmentación de la imagen

Para este primer paso, se elige la imagen, a la cual se le aplicará este algoritmo.

Posteriormente, esta imagen será transformada a binaria, es decir, la imagen tomará valores de "0" y "1", con esto la imagen se verá en blanco y negro, este proceso se hace, porque las memorias asociativas Alfa-Beta trabajan únicamente con valores binarios "0" y "1", con esto ya no tendremos problemas para trabajar con la imagen, esto se puede hacer mediante software, por ejemplo MATLAB, para transformar la imagen a binaria, como se muestra en la Fig. 3.1.

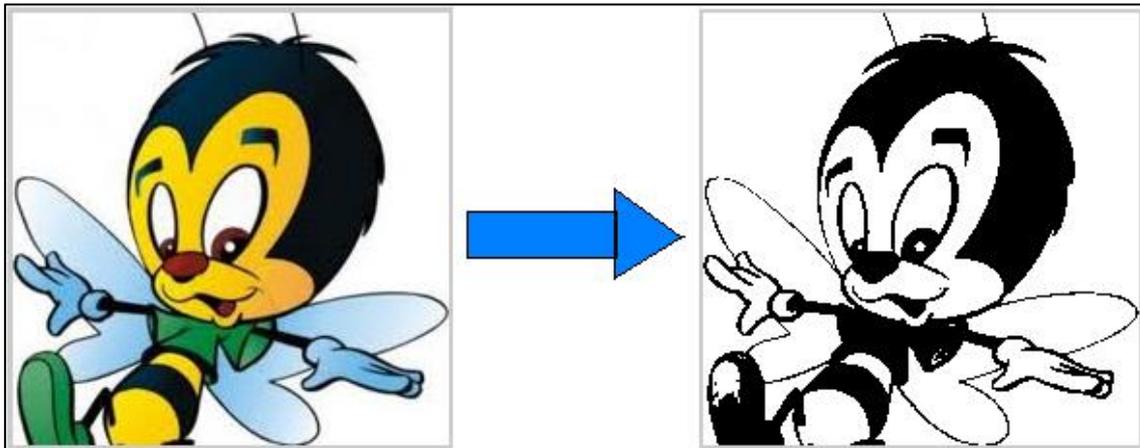


Fig. 3.1 Conversión de una imagen RGB a binaria

Posteriormente, hay que seccionar la imagen en pequeños fragmentos de 3x3 como se ve en la Fig. 3.2, pero se debe considerar, que si la imagen no es múltiplo de 3, a ésta se le agregarán "ceros" o "unos" en el contorno de la imagen, hasta hacerla múltiplo de 3, de lo contrario se deja intacta la imagen.

En caso de que se hayan agregado "ceros" o "unos" a la imagen, al final del algoritmo estos "ceros" o "unos" tendrán que ser eliminados, para obtener el tamaño de la imagen original.

Los fragmentos de 3x3 de la imagen, serán tomados, una vez que la imagen sea binarizada, para ejemplificarlo, podemos verlo en la Fig. 3.2, considerando que el tamaño de la figura es de 6 x 6.

1	1	1	0	0	1
1	1	1	0	0	1
1	1	1	0	0	1
1	1	1	0	0	1
0	0	0	0	0	1
0	0	0	0	0	1

Fig. 3.2 Tomando fragmentos de 3x3, de una imagen de 6x6

3.2 Aplicación del algoritmo, de memorias Alfa-Beta

Después de haber dividido toda la imagen en fragmentos de 3x3, se deberán de pasar, a forma de vector los fragmentos obtenidos, habiendo quedado de la siguiente forma, por ejemplo, tomando el primer fragmento de la Fig. 3.2, y posteriormente pasado a forma de vector, tendremos nuestro vector de la siguiente forma, como lo muestra la Fig. 3.3.

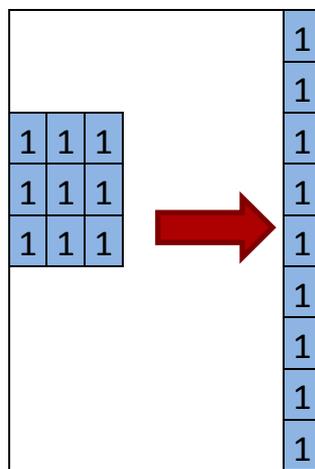


Fig. 3.3 Vector de 9x1

Una vez hecho esto, se obtendrán n vectores de 9x1, dependiendo del tamaño de la imagen, así entonces, obtendremos nuestros patrones de entrada, para así al hacer uso del algoritmo de las memorias asociativas Alfa-Beta.

Para esto, empezamos por operar cada vector con su mismo transpuesto, aplicando la regla de aprendizaje α (Fig. 2.2), expuesta en el capítulo 2, y como resultado se obtendrá una matriz de 9x9, y se hace lo mismo para cada vector obtenido de cada fragmento de la imagen, la Fig. 3.4

representa el resultado correspondiente al primer fragmento de la imagen, para el primer fragmento tenemos:

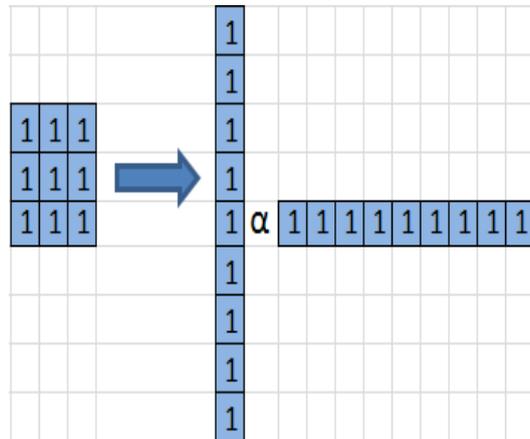


Fig. 3.4 Vector x vector transpuesto, aplicando la regla Alfa

Como se puede ver en la Fig. 3.4, hacemos la operación aplicando la regla Alfa previamente explicada en el capítulo 2 (Fig. 2.2).

Una vez finalizado este paso, se obtienen n matrices de 9x9, las cuales serán comparadas elemento a elemento, aplicándoles el operador de máximos (\vee), y luego el de mínimos (\wedge), que es parte del algoritmo de las memorias Alfa-Beta, descrito en el capítulo 2, así obtendremos 2 matrices: una de máximos y otra de mínimos, ambas de 9x9, podemos ver esto con más detalle en el ejemplo siguiente.

Retomando el ejemplo anterior, donde consideramos que la imagen es de un tamaño de 6x6, obtenemos, la primera matriz 9x9, correspondiente al primer fragmento de la imagen, como lo muestra la Fig. 3.5:

Capítulo 3 Desarrollo

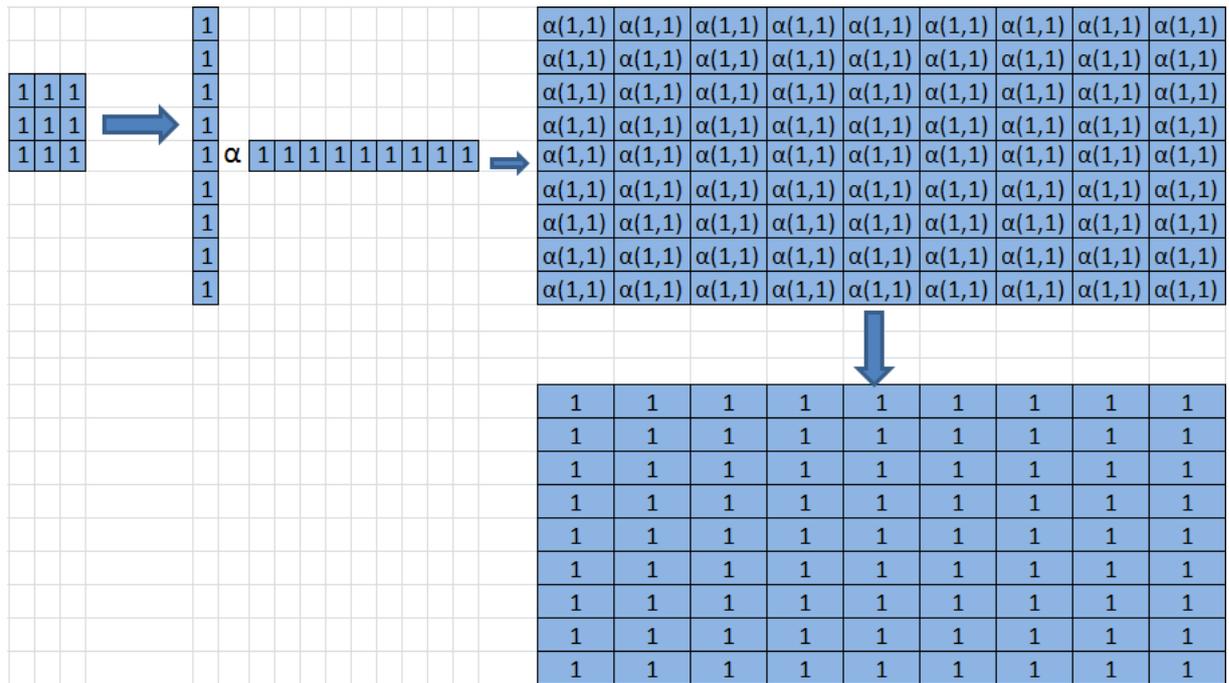


Fig. 3.5 Matriz del primer fragmento de la imagen

Obtenemos la segunda matriz, correspondiente al segundo fragmento de la imagen leída, aplicando la regla de aprendizaje de las memorias Alfa-Beta (ver Fig. 2.2), quedándonos la siguiente matriz, como lo muestra la Fig. 3.6.

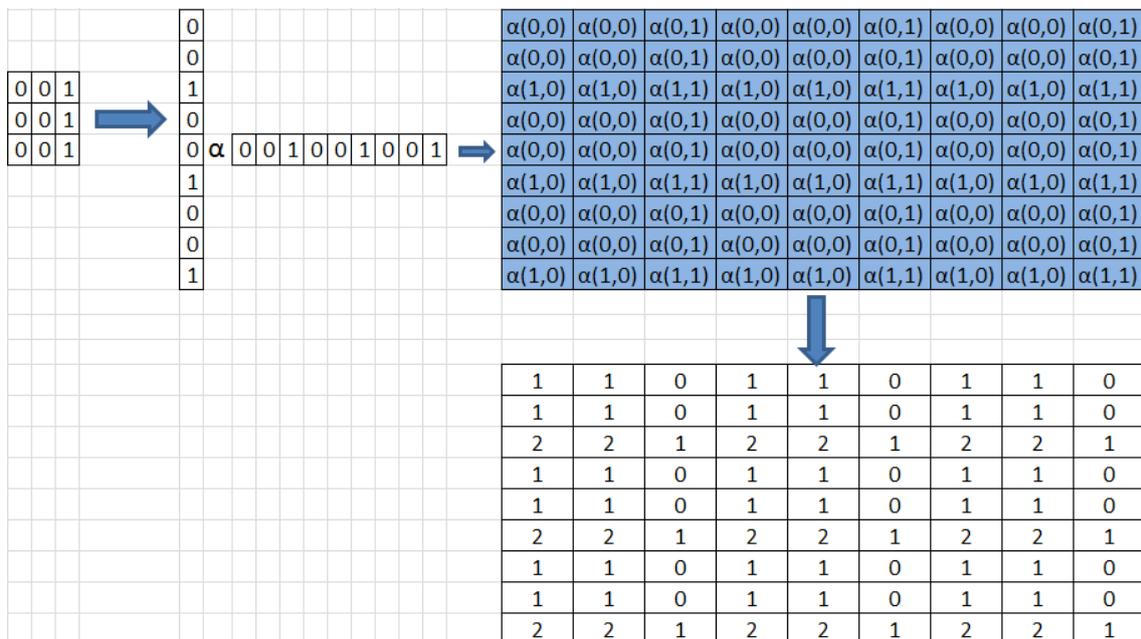


Fig. 3.6 Matriz del segundo fragmento de la imagen

Capítulo 3 Desarrollo

Obtenemos la tercera matriz, correspondiente al tercer fragmento de la imagen, aplicando la regla de aprendizaje de las memorias Alfa-Beta (ver Fig. 2.2), como se ve en la Fig. 3.7.

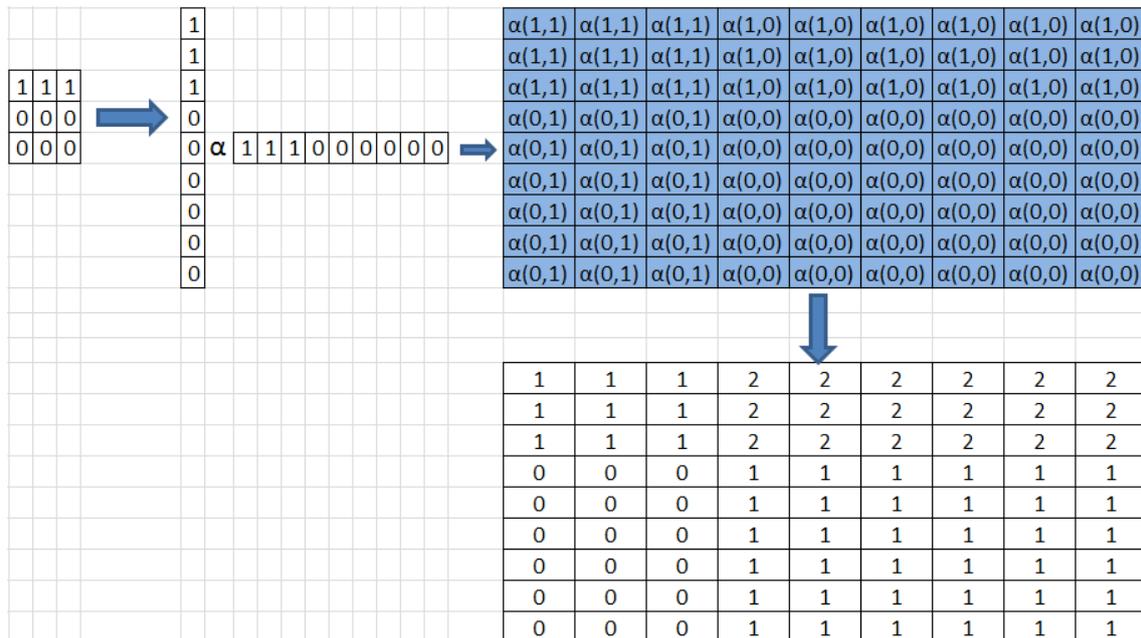


Fig. 3.7 Matriz del tercer fragmento de la imagen

Obtenemos la cuarta matriz, correspondiente al cuarto fragmento de la imagen, aplicando la regla de aprendizaje de las memorias Alfa-Beta (ver Fig. 2.2), como lo vemos en la Fig. 3.8.

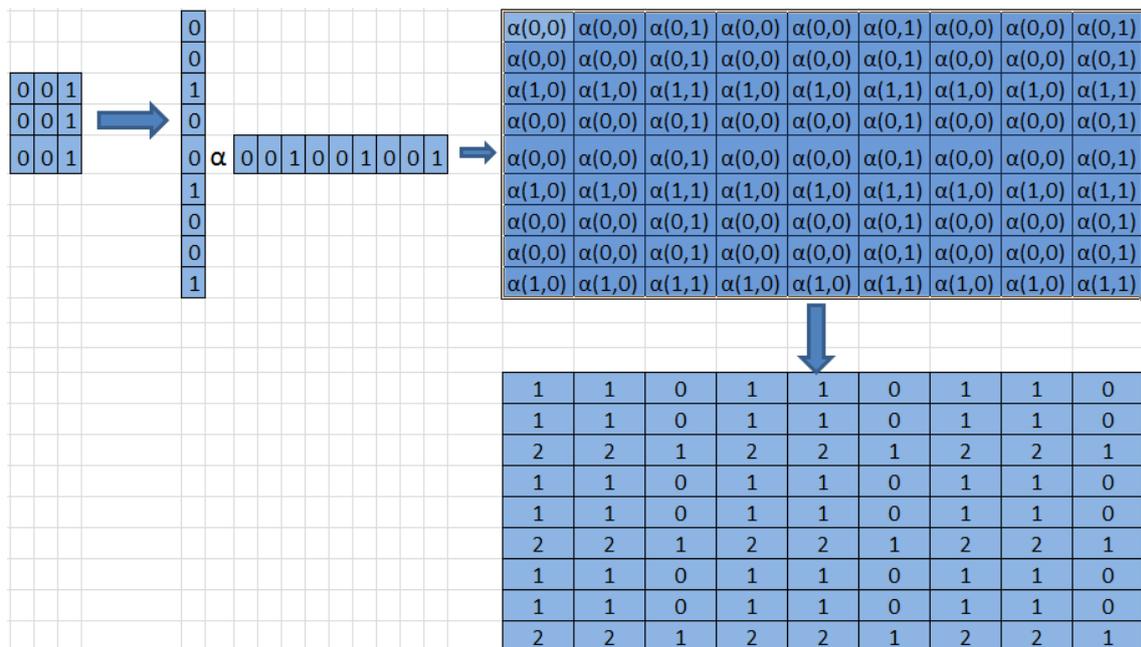


Fig. 3.8 Matriz del cuarto fragmento de la imagen

De esta manera, ya obtenidas todas nuestras matrices de 9x9, enseguida comparamos todas las matrices obtenidas, aplicando el operador de

Capítulo 3 Desarrollo

Hay que recordar que para poder obtener los eigenvectores de una matriz, primero se necesitan calcular sus eigenvalores, como lo muestra la Fig. 3.10 y la Fig. 3.12.

16,31357422	0	0	0	0	0	0	0	0	0
0	-0,609050435	0	0	0	0	0	0	0	0
0	0	-0,704523784	0	0	0	0	0	0	0
0	0	0	-1,000000014	0	0	0	0	0	0
0	0	0	0	-0,999999986	0	0	0	0	0
0	0	0	0	0	-1	0	0	0	0
0	0	0	0	0	0	-1	5,13E-09	0	0
0	0	0	0	0	0	-5,13E-09	-1	0	0
0	0	0	0	0	0	0	0	0	-1

Fig. 3.10 Eigenvalores para la matriz de máximos (M)

La Fig. 3.11, muestra los eigenvectores obtenidos, de la matriz de máximos, después de haber calculado sus eigenvalores (Fig. 3.10).

0,326281398	0,351905127	0,310254134	0,399465665	0,399465488	-0,562474201	-0,455147948	-1,04E-07	0,470623968
0,346282006	-0,225887953	-0,130120117	-4,66E-08	4,66E-08	-3,00E-13	-4,69E-13	1,51E-07	-0,019134781
0,307435988	-0,548224094	-0,739759765	-0,82473546	-0,824735537	-0,260085526	0,747860591	0	-0,756050451
0,346282006	-0,225887953	-0,130120117	5,46E-09	-5,46E-09	1,03E-15	7,62E-15	2,34E-09	1,83E-14
0,346282006	-0,225887953	-0,130120117	3,53E-10	-3,53E-10	3,73E-16	-4,07E-15	-8,34E-10	-4,91E-15
0,326281398	0,351905127	0,310254134	-1,13E-08	1,13E-08	1,07E-16	-1,15E-14	-3,84E-09	-3,12E-14
0,326281398	0,351905127	0,310254134	0,025804275	0,025804416	0,038670121	0,162435304	-2,18E-07	-0,130699823
0,326281398	0,351905127	0,310254134	0,399465572	0,399465581	0,783889606	-0,455147948	1,73E-07	0,435261087
0,346282006	-0,225887953	-0,130120117	-2,15E-17	-3,10E-17	-2,22E-16	-1,06E-16	-1,67E-23	1,59E-16

Fig. 3.11 Eigenvectores para la matriz de máximos (M)

La Fig. 3.12, muestra los eigenvalores correspondientes a la matriz de mínimos.

1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1

Fig. 3.12 Eigenvalores para la matriz de mínimos (W)

La Fig. 3.13, muestra los eigenvectores correspondientes a la matriz de mínimos, después de haber calculado sus eigenvectores (Fig. 3.12).

0	2,22E-16	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	4,93E-32	0	0	0	0
1	-1	0	0	0	0	0	0	0	-1
0	0	0	0	0	0	1	-1	0	0
0	0	0	0	2,22E-16	-2,22E-16	0	0	0	0
0	0	0	0	0	0	0	2,22E-16	0	0
0	0	0	0	0	0	0	0	2,22E-16	0
0	0	0	1	-1	1	0	0	0	0

Fig. 3.13 Eigenvectores para la matriz de mínimos (W)

Los eigenvectores obtenidos aquí, tanto para la matriz de máximos como la de mínimos, se encuentran ubicados en forma de vector de 9x1, como se muestra en la Fig. 3.11 y Fig. 3.13, cada columna corresponde a un eigenvector.

Una vez hecho esto, se deberán convertir los eigenvectores obtenidos, que tienen un tamaño de 9x1, a una matriz de 3x3, como se muestra a continuación en la Fig. 3.14, para que posteriormente sean utilizados como máscaras, para ser aplicados a la imagen, como se muestra a continuación.

Por ejemplo, para el primer eigenvector de la matriz de máximos, tenemos:

0,326281398				
0,346282006				
0,307435988				
0,346282006		0,326281398	0,346282006	0,307435988
0,346282006		0,346282006	0,346282006	0,326281398
0,326281398		0,326281398	0,326281398	0,346282006
0,326281398				
0,326281398				
0,346282006				

Fig. 3.14 Primer eigenvector obtenido de la matriz de máximos, convertido a matriz de 3x3

Hacemos lo mismo para cada eigenvector, correspondiente a la matriz de mínimos, y también para cada eigenvector correspondiente a la matriz de máximos.

Una vez convertidos a forma de matriz, los 18 eigenvectores, por consiguiente surgirán 18 matrices de 3x3, recordando que 9 matrices pertenecen a mínimos, y las otras 9 pertenecen a máximos.

3.4 Convolución de máscaras

Ahora como siguiente paso, hay que aplicar cada máscara (matriz de 3x3) a la imagen, esto se logra convolucionando la matriz de 3x3 (Fig. 3.14) obtenida, con la imagen seleccionada, de la misma forma que los algoritmos ya definidos, como lo son el de: Roberts, Sobel, Canny, etc., así se observará cuál máscara brinda el mejor resultado.

Como ya se ha mencionado anteriormente, hay que convolucionar las máscaras obtenidas con la imagen, para así poder decidir cuál máscara es la más adecuada para este algoritmo.

Ejemplo 1

La Fig. 3.15, representa la imagen original.

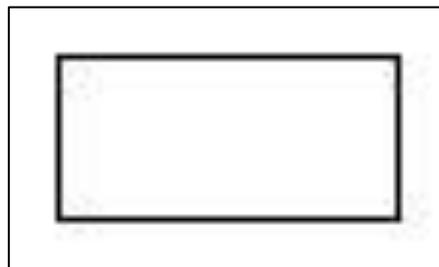


Fig. 3.15 Rectángulo

La Fig. 3.16 muestra los resultados obtenidos con los 9 eigenvectores correspondientes a la matriz de mínimos, aplicados a la Fig. 3.15.

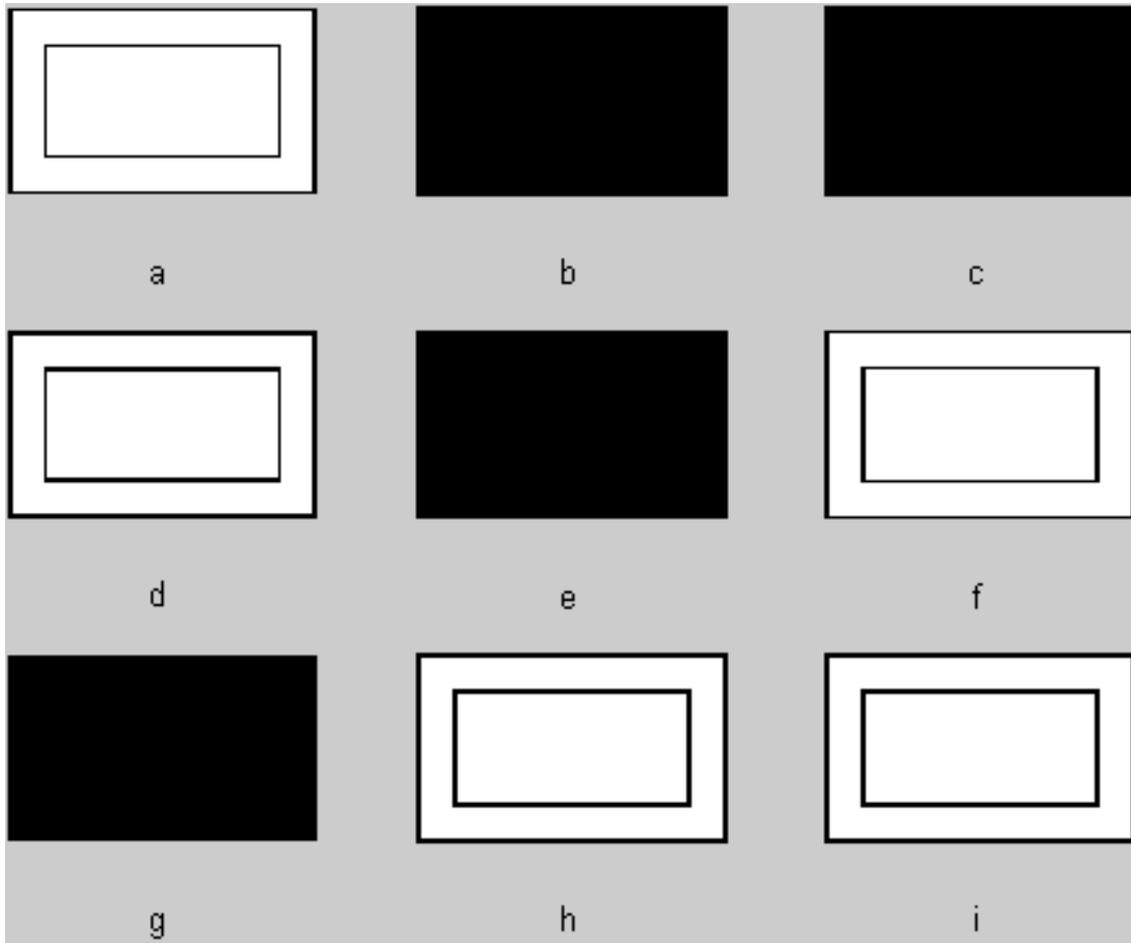


Fig. 3.16 Eigenectores correspondientes a mínimos, aplicados como máscaras al ejemplo 1

La letra "a" corresponde al eigenvector 1 para mínimos, la letra "b" corresponde al eigenvector 2 para mínimos, la letra "c" corresponde al eigenvector 3 para mínimos, y así sucesivamente para los nueve eigenvectores.

Con esto, podemos ver resultados no deseados, obtenidos con los eigenvectores de la matriz que corresponde a mínimos, que no son lo que esperamos.

La Fig. 3.17 muestra los resultados obtenidos con los 9 eigenvectores correspondientes a la matriz de máximos, aplicados a la Fig. 3.15.

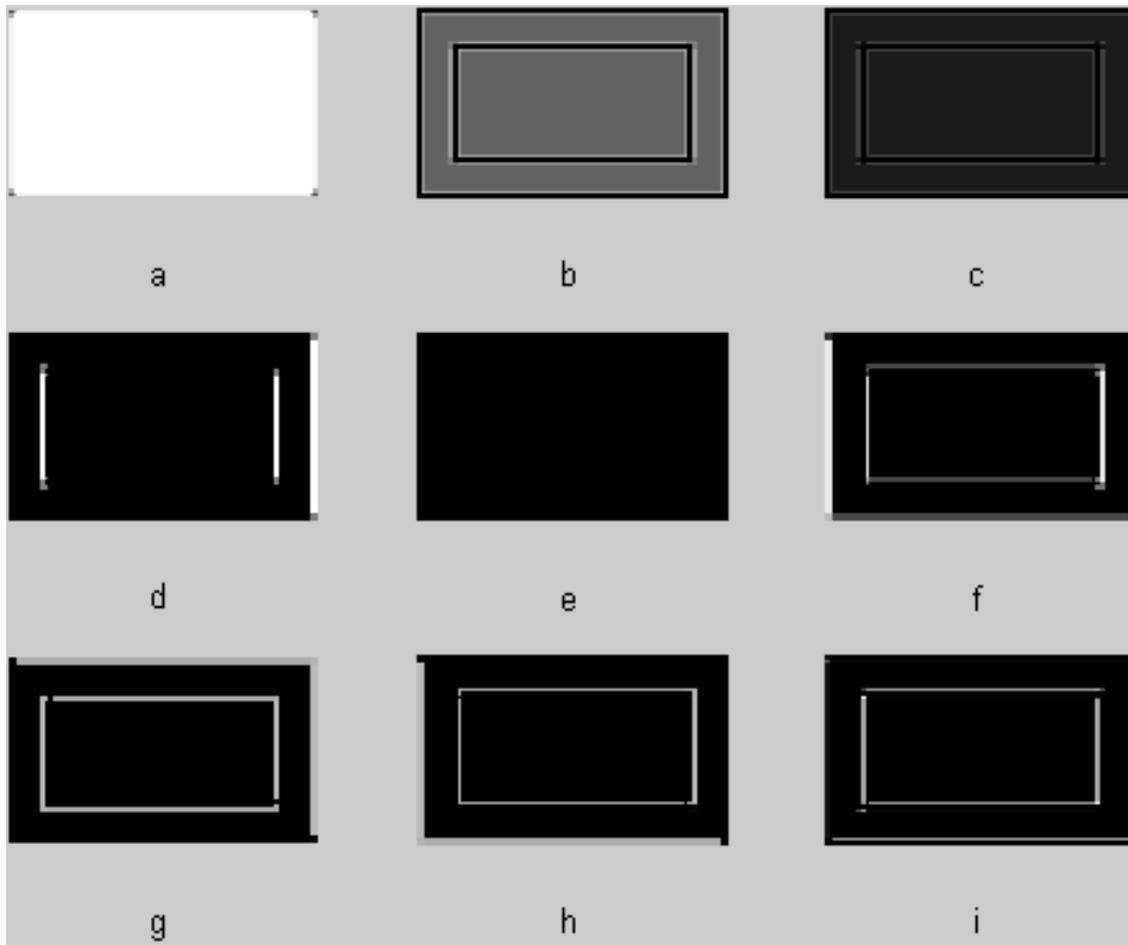


Fig. 3.17 Eigenectores correspondientes a máximos, aplicados como máscaras al ejemplo 1

La letra "a" corresponde al eigenvector 1 para máximos, la letra "b" corresponde al eigenvector 2 para máximos, la letra "c" corresponde al eigenvector 3 para máximos, y así sucesivamente para los nueve eigenvectores.

Con esto, podemos ver mejores resultados, con los eigenvectores correspondientes a la matriz de máximos, aplicados a la imagen como máscaras.

Ejemplo 2

La Fig. 3.18 representa la imagen original.

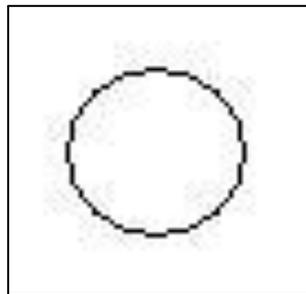


Fig. 3.18 Círculo

La Fig. 3.19 muestra los resultados obtenidos con los 9 eigenvectores correspondientes a la matriz de mínimos, aplicados a la Fig. 3.18.

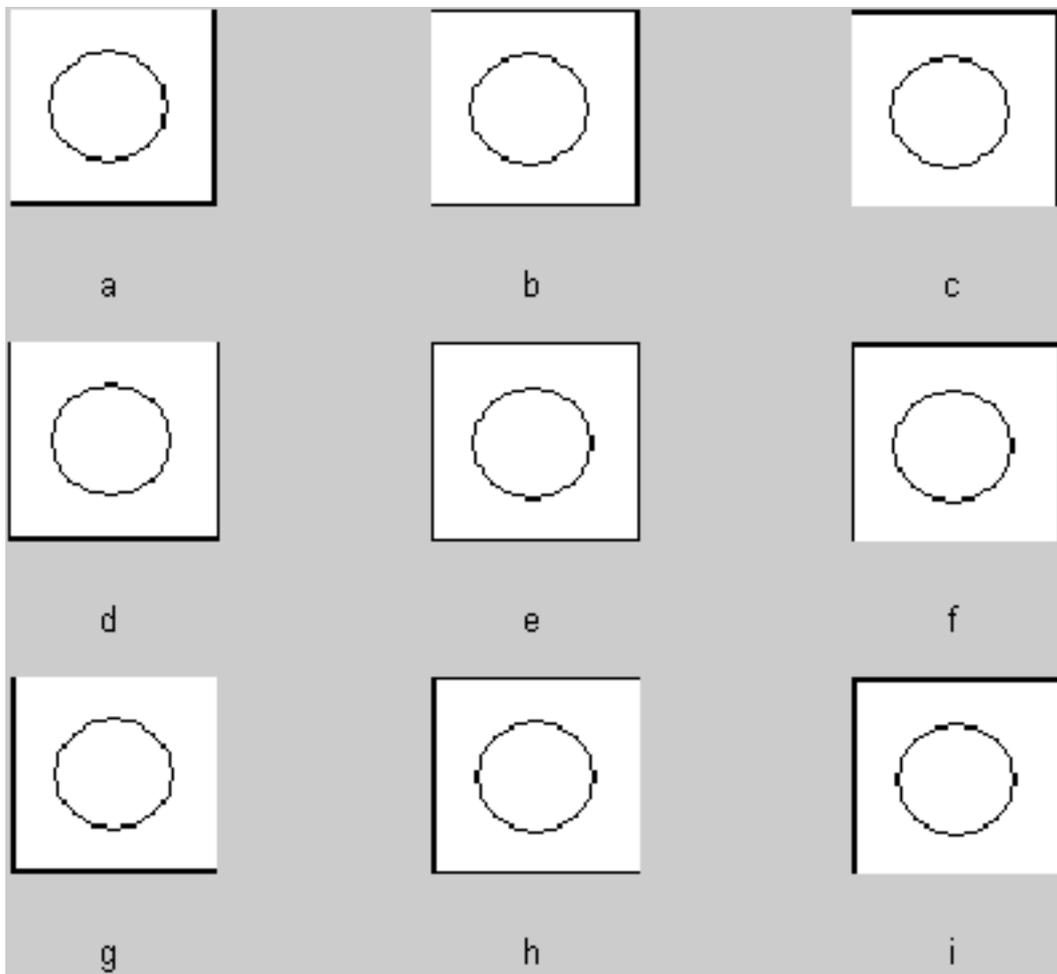


Fig. 3. 19 Eigenvectores correspondientes a mínimos, aplicados como máscaras al ejemplo 2

La Fig. 3.20 muestra los resultados obtenidos con los 9 eigenvectores correspondientes a la matriz de máximos, aplicados a la Fig. 3.18.

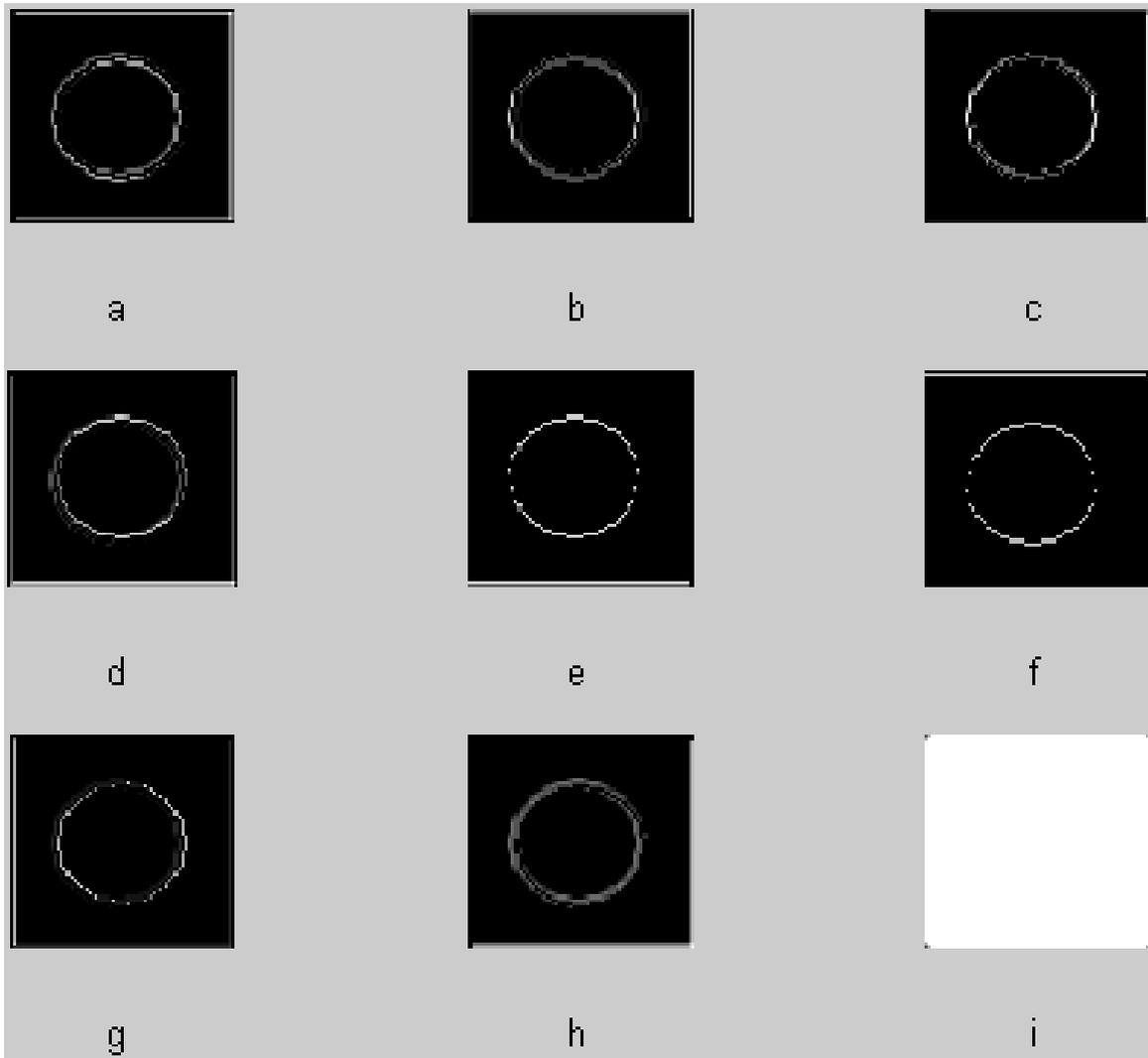


Fig. 3. 20 Eigenvectores correspondientes a máximos, aplicados como máscaras al ejemplo 2

Como se puede observar, los resultados obtenidos son parecidos a los obtenidos en el ejemplo 1, y al igual que en el ejemplo anterior, cada letra corresponde a un eigenvector: "a" corresponde al eigenvector 1, "b" corresponde al eigenvector 2, y así sucesivamente.

Ejemplo 3

La Fig. 3.21 representa la imagen original.



Fig. 3.21 Imagen de Lena

La Fig. 3.22 muestra los resultados obtenidos con los 9 eigenvectores correspondientes a la matriz de mínimos, aplicados a la Fig. 3.21.

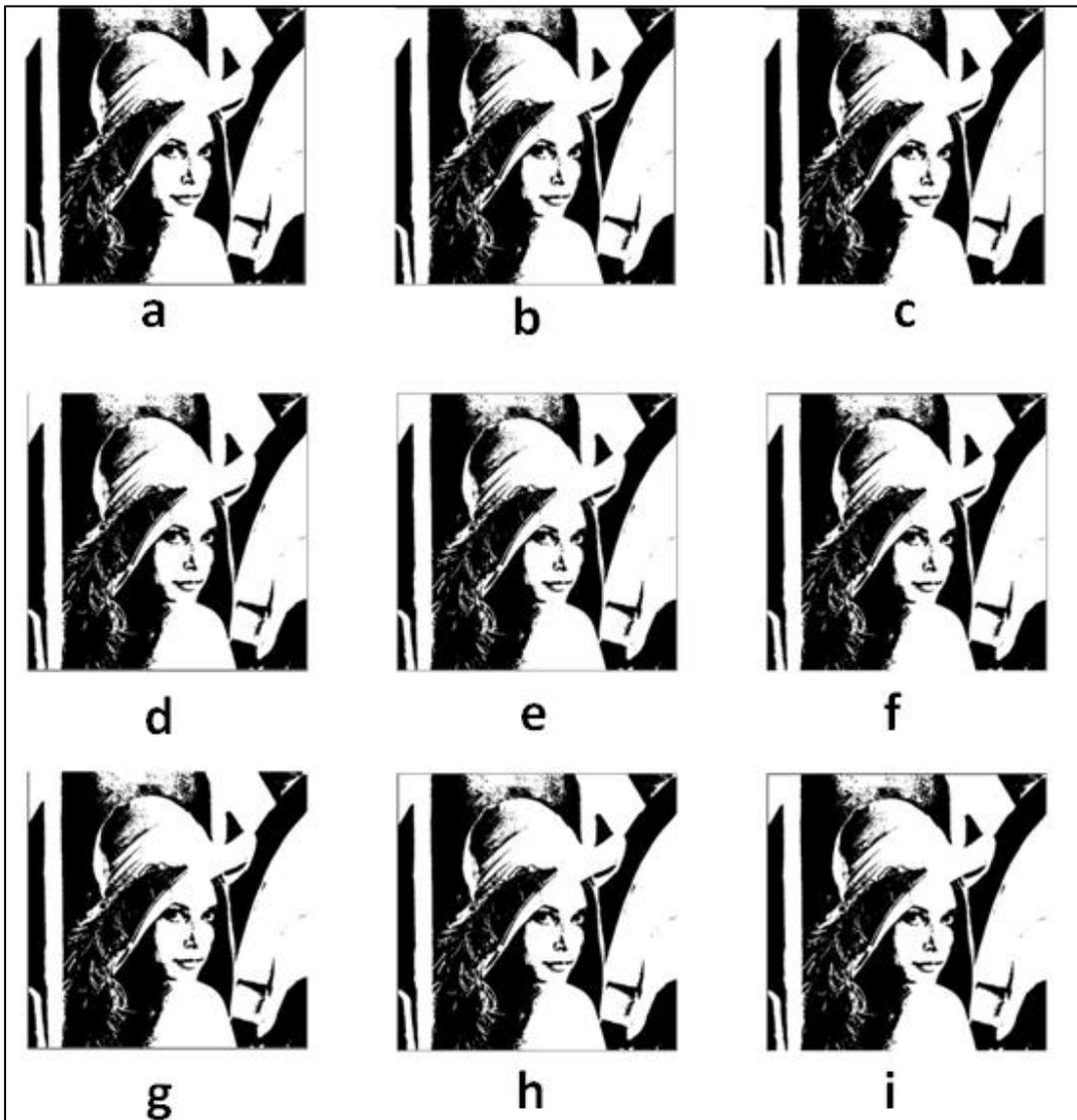


Fig. 3.22 Eigenvectores correspondientes a mínimos, aplicados como máscaras al ejemplo 3

La Fig. 3.23 muestra los resultados obtenidos con los 9 eigenvectores correspondientes a la matriz de máximos, aplicados a la figura 3.21.

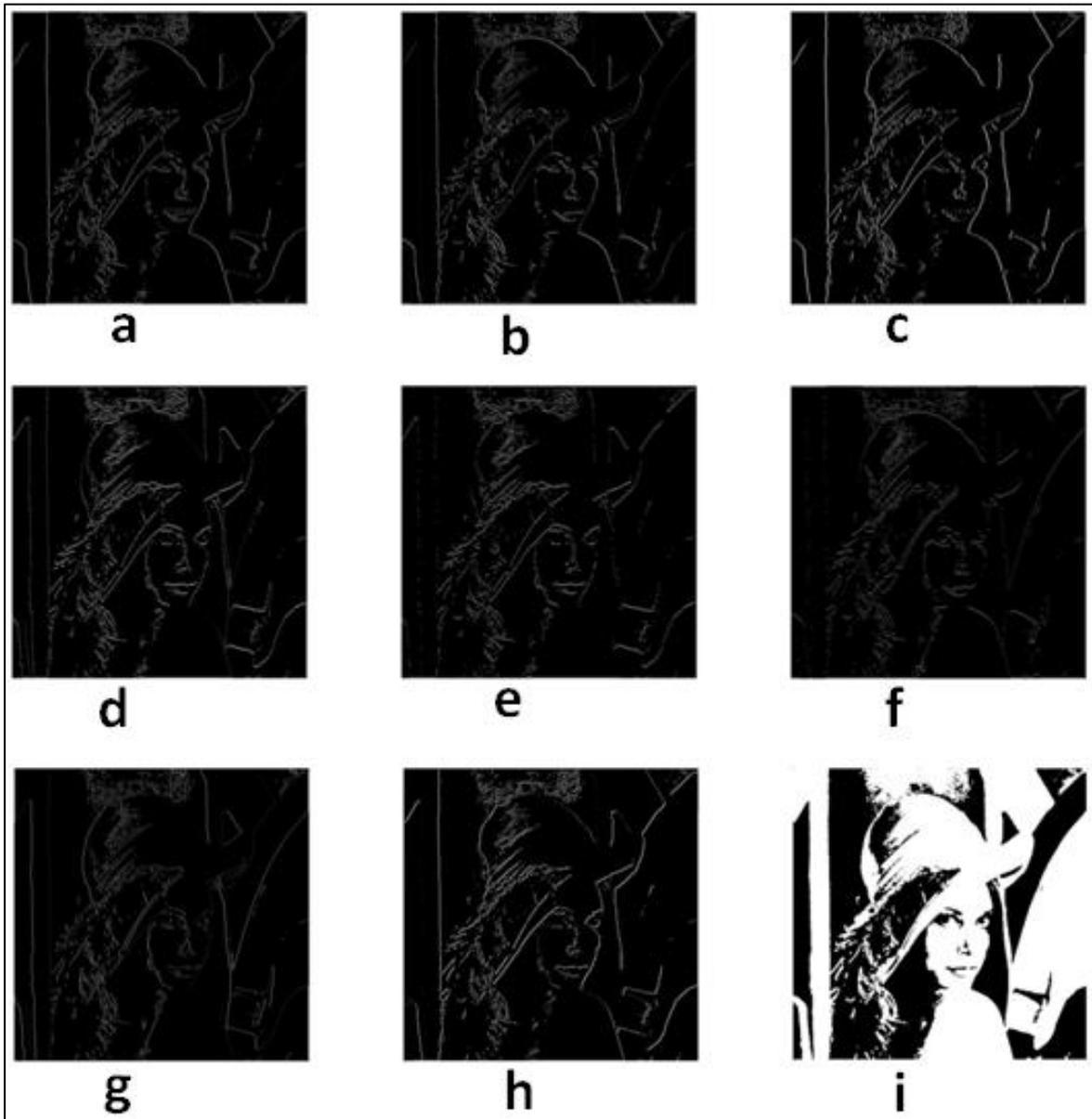


Fig. 3.23 Eigenvectores correspondientes a máximos, aplicados como máscaras al ejemplo 3

El inciso "a" corresponde al eigenvector 1, "b" corresponde al eigenvector 2, y así sucesivamente para ambos casos: máximos y mínimos.

Ejemplo 4

La Fig. 3.24 representa la imagen original.



Fig. 3.24 Abeja

La Fig. 3.25 muestra los resultados obtenidos con los 9 eigenvectores correspondientes a la matriz de mínimos, aplicados a la Fig. 3.24.

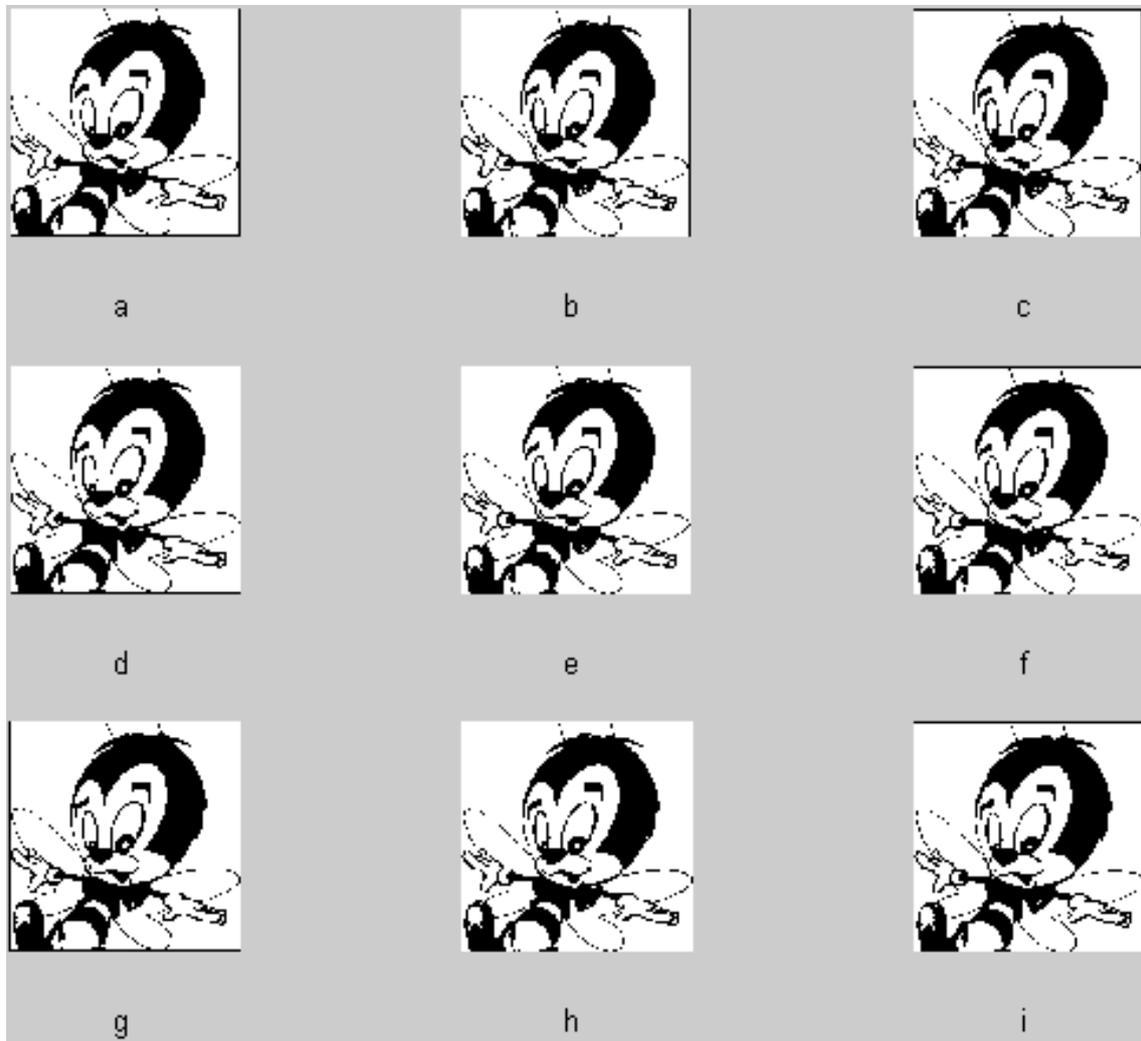


Fig. 3.25 Eigenvectores correspondientes a mínimos, aplicados como máscaras al ejemplo 4

La Fig. 3.26 muestra los resultados obtenidos con los 9 eigenvectores, correspondientes a la matriz de máximos, aplicados a la figura 3.24.

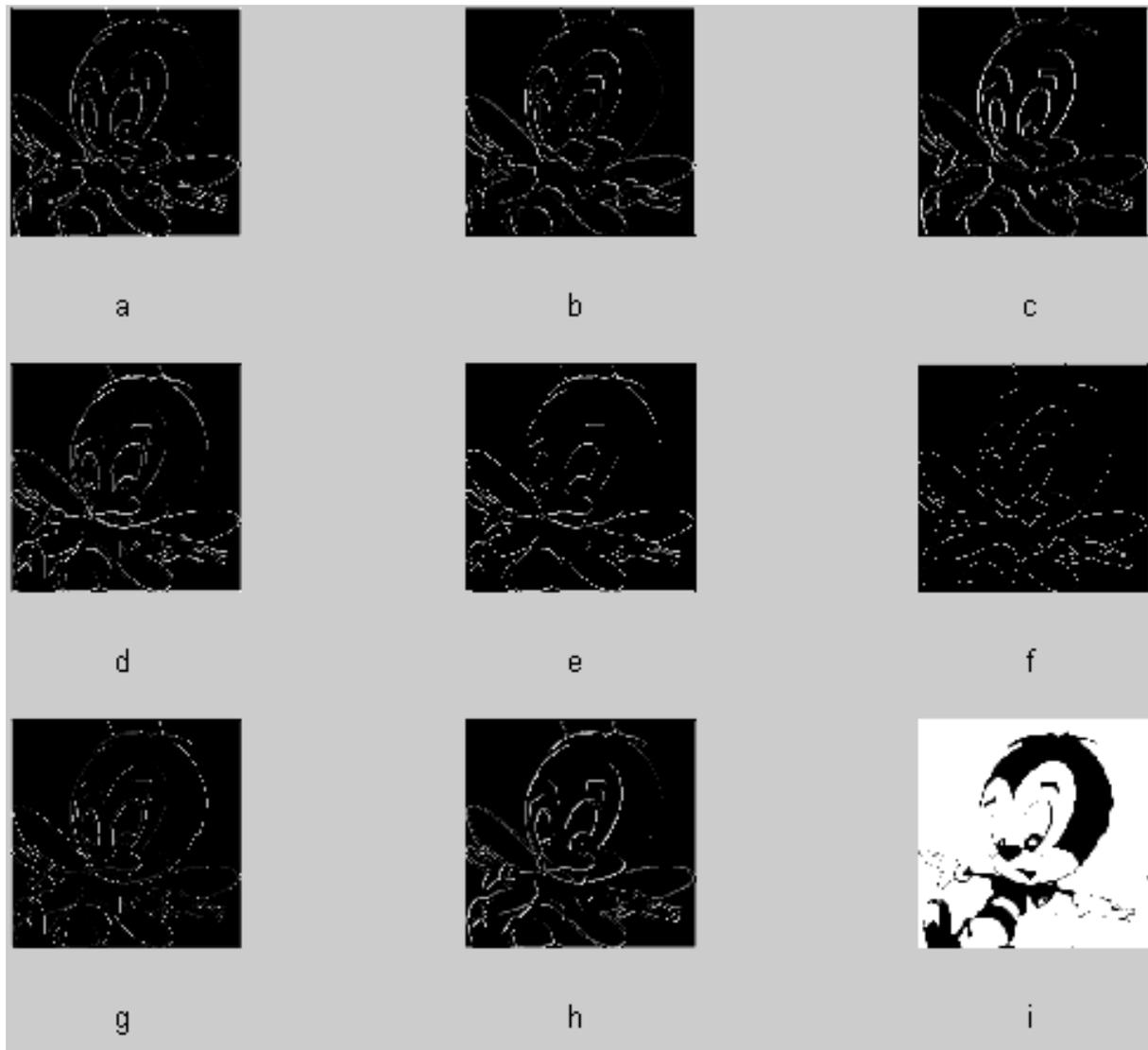


Fig. 3.26 Eigenectores correspondientes a máximos, aplicados como máscaras al ejemplo 4

El inciso "a" corresponde al eigenvector 1, "b" corresponde al eigenvector 2, y así sucesivamente para ambos casos: máximos y mínimos.

Ejemplo 5

La Fig. 3.27 representa la imagen original.



Fig. 3.27 Gato Félix

La Fig. 3.28 muestra los resultados obtenidos con los 9 eigenvectores correspondientes a la matriz de mínimos, aplicados a la Fig. 3.27.



Fig. 3.28 Eigenvectores correspondientes a mínimos, aplicados como máscaras al ejemplo 5

La Fig. 3.29 muestra los resultados obtenidos con los 9 eigenvectores, correspondientes a la matriz de máximos, aplicados a la figura 3.27.

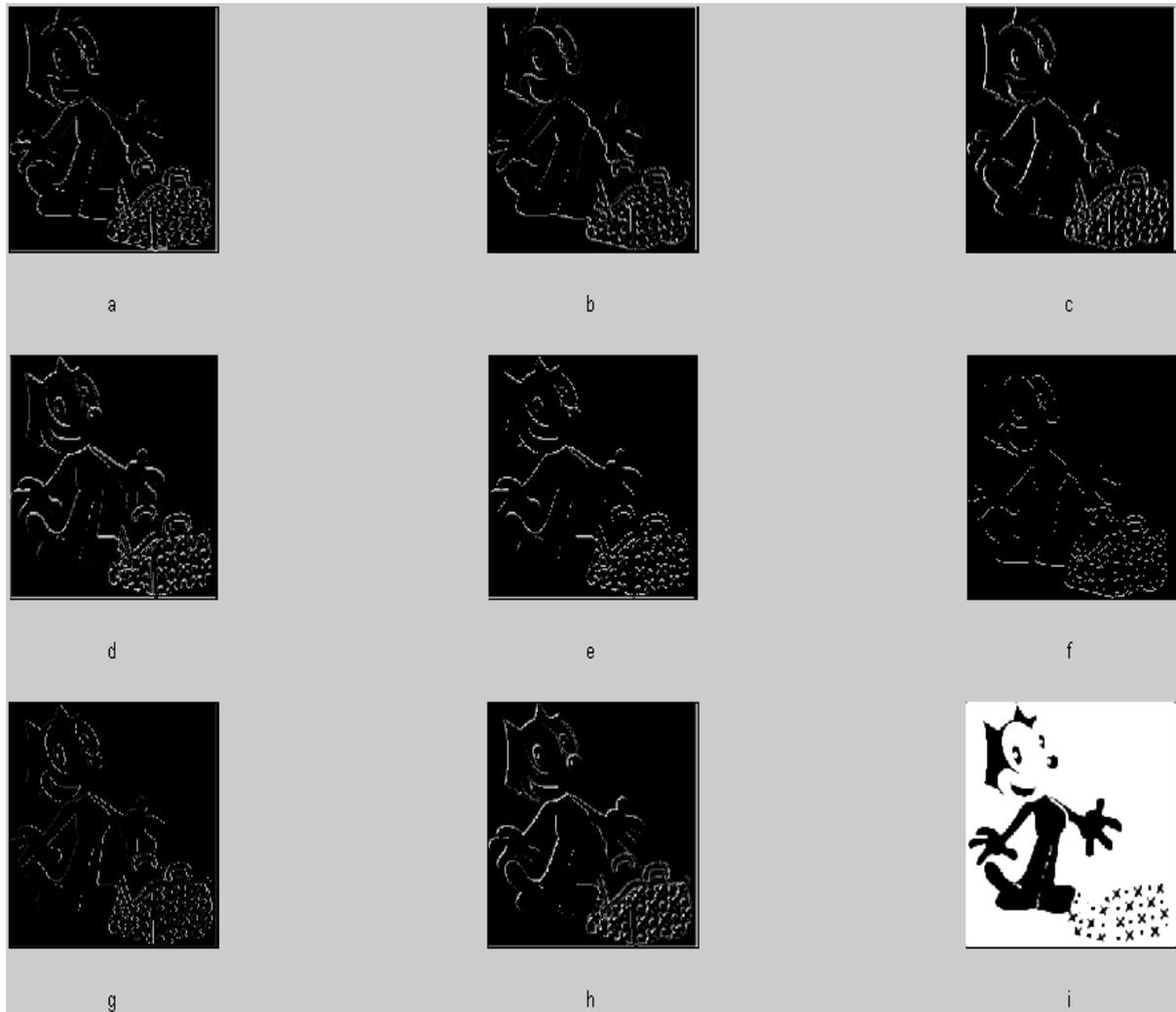


Fig. 3.29 Eigenectores correspondientes a máximos, aplicados como máscaras al ejemplo 5

El inciso "a" corresponde al eigenvector 1, "b" corresponde al eigenvector 2, y así sucesivamente para ambos casos: máximos y mínimos.

Ejemplo 6

La Fig. 3.30 representa la imagen original.



Fig. 3.30 Gato

La Fig. 3.31 muestra los resultados obtenidos con los 9 eigenvectores correspondientes a la matriz de mínimos, aplicados a la Fig. 3.30.

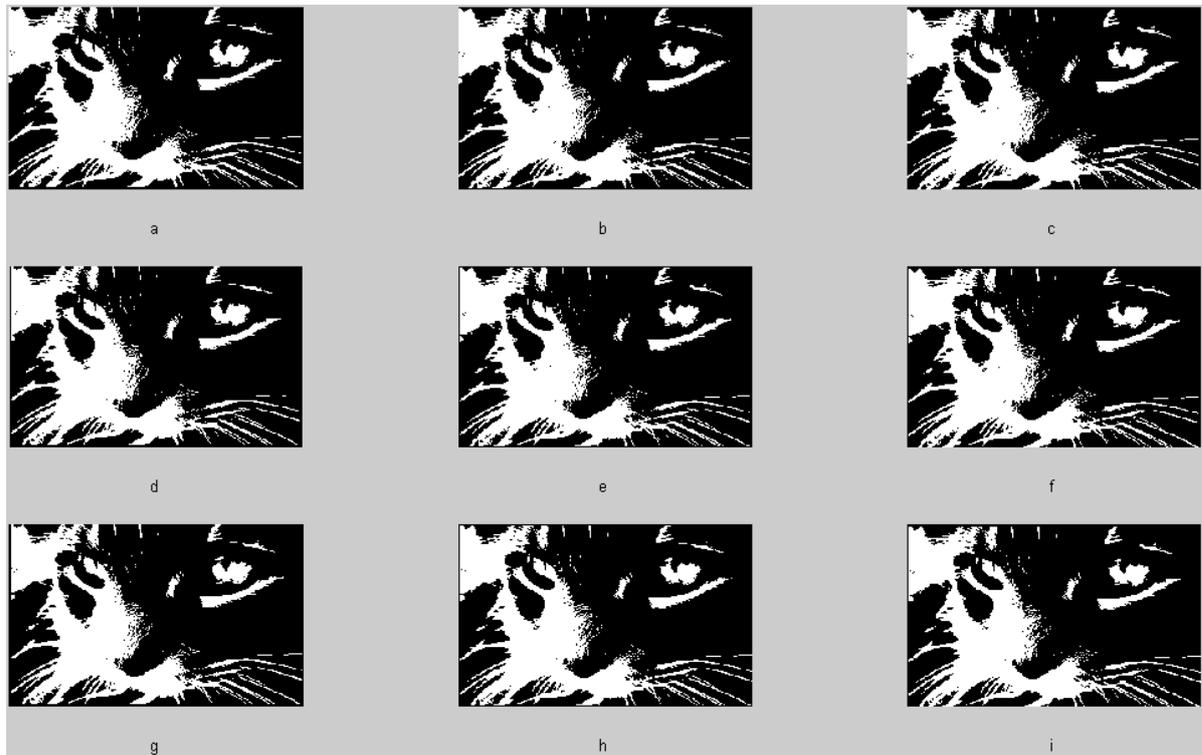


Fig. 3.31 Eigenvectores correspondientes a mínimos, aplicados como máscaras al ejemplo 6

La Fig. 3.32 muestra los resultados obtenidos con los 9 eigenvectores correspondientes a la matriz de máximos, aplicados a la figura 3.30.

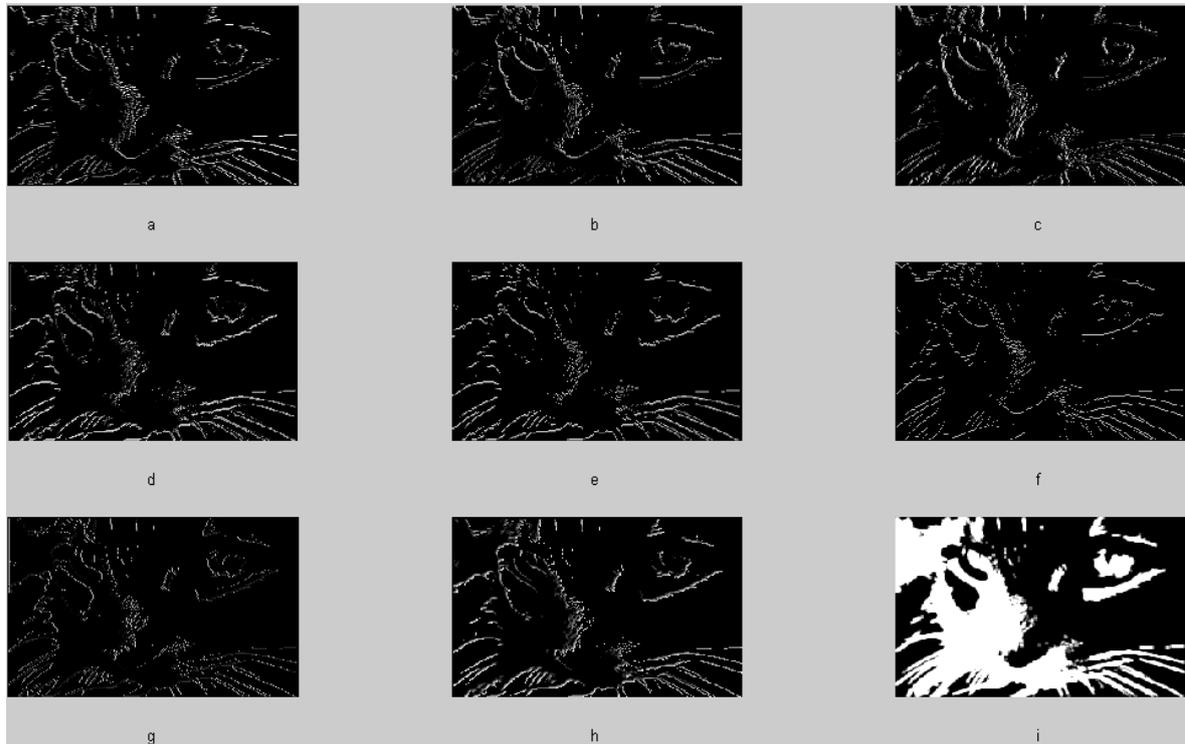


Fig. 3.32 Eigenvectores correspondientes a máximos, aplicados como máscaras al ejemplo 6

El inciso "a" corresponde al eigenvector 1, "b" corresponde al eigenvector 2, y así sucesivamente para ambos casos: máximos y mínimos.

Ejemplo 7

La Fig. 3.33 representa la imagen original.



Fig. 3.33 Sombras

La Fig. 3.34 muestra los resultados obtenidos con los 9 eigenvectores, correspondientes a la matriz de mínimos, aplicados a la Fig. 3.33.

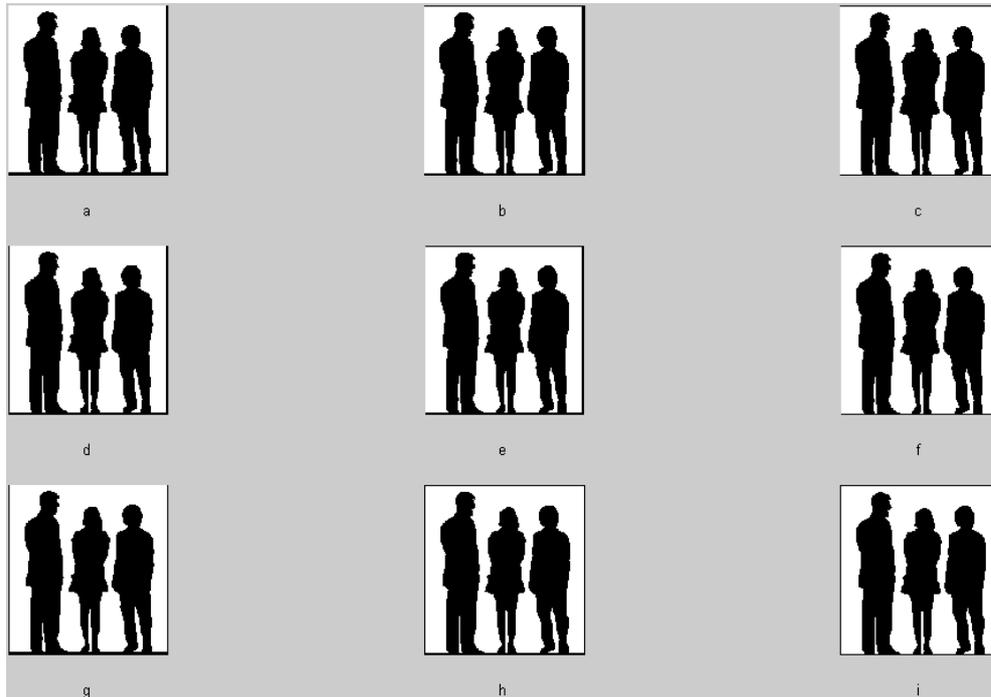


Fig. 3.34 Eigenvectores correspondientes a mínimos, aplicados como máscaras al ejemplo 7

La Fig. 3.35 muestra los resultados obtenidos con los 9 eigenvectores, correspondientes a la matriz de máximos, aplicados a la figura 3.33.

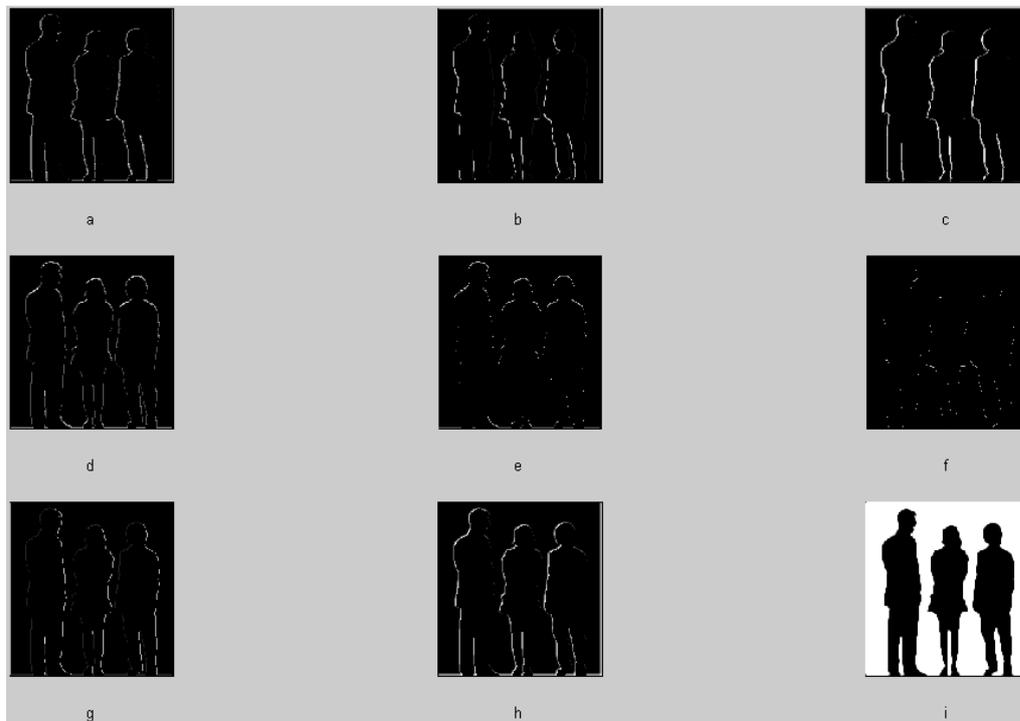


Fig. 3.35 Eigenvectores correspondientes a máximos, aplicados como máscaras al ejemplo 7

El inciso "a" corresponde al eigenvector 1, "b" corresponde al eigenvector 2, y así sucesivamente para ambos casos: máximos y mínimos.

Lo mismo aplica para el ejemplo 8.

Ejemplo 8

La Fig. 3.36 representa la imagen original.



Fig. 3.36 Dibujo animado

La Fig. 3.37 muestra los resultados obtenidos con los 9 eigenvectores, correspondientes a la matriz de mínimos, aplicados a la Fig. 3.36.

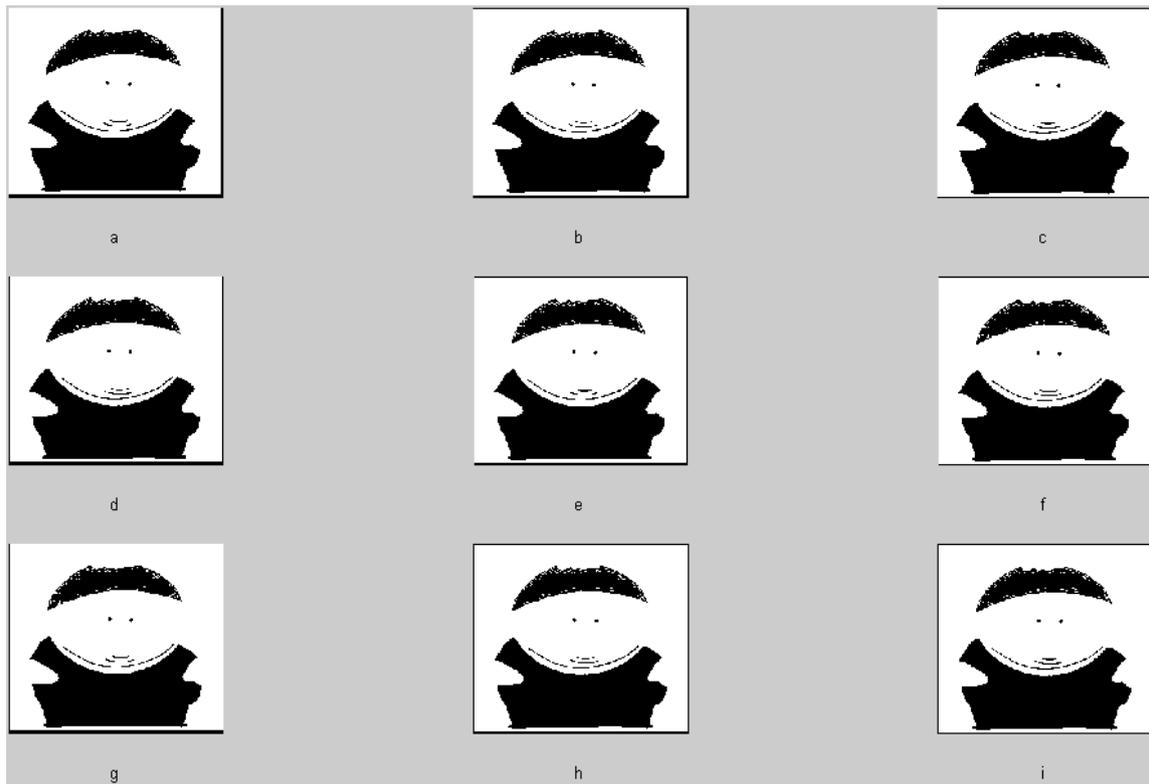


Fig. 3.37 Eigenvectores correspondientes a mínimos, aplicados como máscaras al ejemplo 8

La Fig. 3.38 muestra los resultados obtenidos con los 9 eigenvectores, correspondientes a la matriz de máximos, aplicados a la figura 3.33.

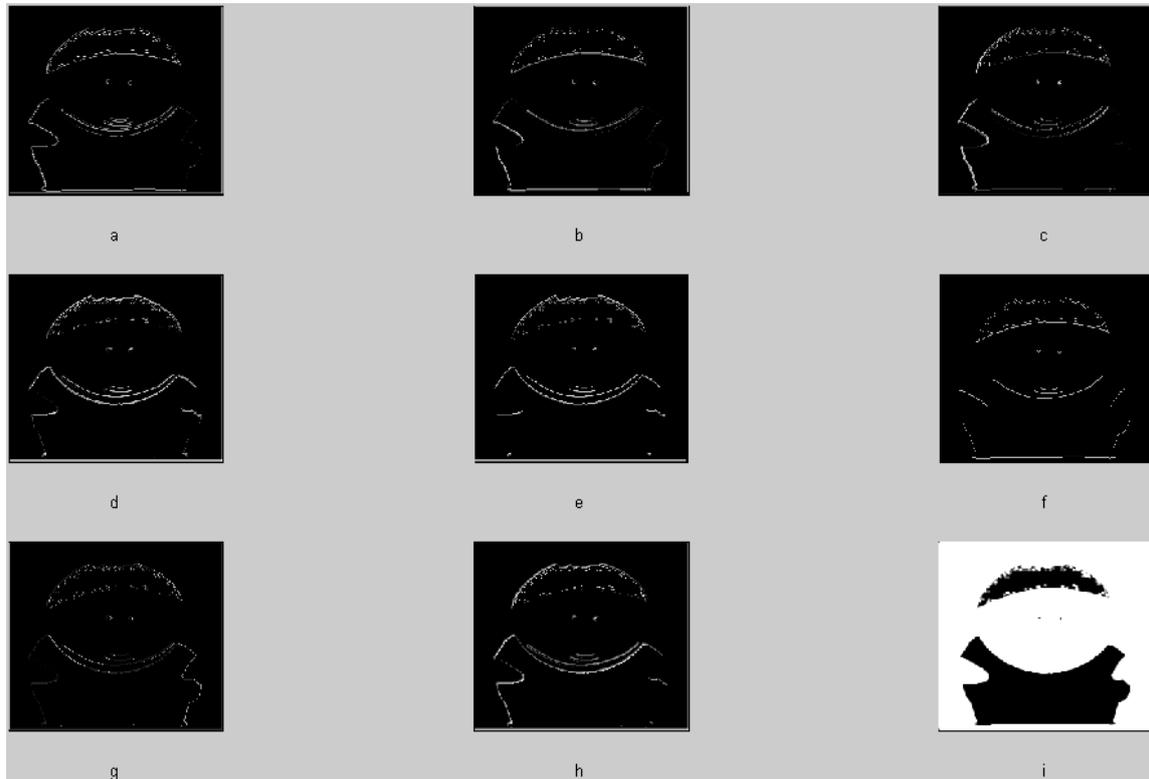


Fig. 3.38 Eigenvectores correspondientes a máximos, aplicados como máscaras al ejemplo 8

Con este último ejemplo, se puede observar, que efectivamente los eigenvectores correspondientes a la matriz formada por mínimos, no sirven para este algoritmo, y por otra parte, que los eigenvectores formados por máximos, arrojan los bordes de la imagen, que es lo que se espera, por lo tanto, se descartan los 9 eigenvectores correspondientes a la matriz de mínimos.

3.5 Selección de eigenvectores

Ya siendo descartados los 9 eigenvectores correspondientes a mínimos, se puede observar, que de los 9 eigenvectores correspondientes a máximos, sobresalen los eigenvectores 4, 6, 7 y 8, correspondientes a las máscaras 4, 6, 7 y 8 de la matriz de máximos, que habían sido creadas anteriormente (Fig. 3.14).

Al aplicar estas máscaras a la imagen, se puede decir que son las que mejores resultados arrojan.

Una vez hecho esto, se hace otra depuración, con lo cual se observa que los eigenvectores 7 y 8 son los que más sobresalen, por arrojar mejores resultados que los eigenvectores 4 y 6.

Para este algoritmo se utiliza el eigenvector 7, que es, el que mejor resultados arroja, esto se puede ver con más detalle en el capítulo 4.

3.6 Aplicación de la máscara a la imagen

El siguiente y último paso de este nuevo método, consiste en obtener una segunda máscara del mismo eigenvector, esto se logra transponiendo el eigenvector seleccionado en forma de matriz (Fig. 3.39), con esto se obtienen dos matrices de 3x3, las cuales serán las máscaras finales para aplicar a la imagen seleccionada.

Siguiendo con el ejemplo anterior, donde consideramos una imagen de 6x6, tenemos lo siguiente.

De la Fig. 3.11, recordando que tenemos 9 eigenvectores ordenados a manera de columnas en la matriz de máximos (M), es decir, cada columna es un eigenvector.

Por lo tanto, tomamos la columna 7 (eigenvector 7), para hacer nuestra segunda máscara, como se aprecia en la Fig. 3.39.

-0,455147948								
-4,69E-13								
0,747860591								
7,62E-15	→	-0,455147948	-4,69E-13	0,747860591		-0,455147948	7,62E-15	0,162435304
-4,07E-15		7,62E-15	-4,07E-15	-1,15E-14		-4,69E-13	-4,07E-15	-0,455147948
-1,15E-14		0,162435304	-0,455147948	-1,06E-16		0,747860591	-1,15E-14	-1,06E-16
0,162435304		mascara 7				mascara 7 transpuesta		
-0,455147948								
-1,06E-16								

Fig. 3.39 Máscaras obtenidas, correspondientes al eigenvector 7

La primera máscara (matriz correspondiente al eigenvector 7), se aplica a la imagen, para obtener los bordes verticales de la imagen, y segunda máscara (matriz transpuesta correspondiente al eigenvector 7), se aplica a la imagen para obtener los bordes horizontales de la imagen.

Para poder aplicar las máscaras a la imagen, hay que recordar que se tiene que convolucionar la imagen con la máscara (matriz).

Capítulo 3 Desarrollo

Una vez hecho esto, se suman ambos resultados obtenidos, después de haber aplicado las máscaras a la imagen.

Para sumar las imágenes, se toman sus valores absolutos, para así tener una mejor imagen.

Después de esto se aplica un umbral, para poder manipular los puntos de bordes, que podemos variar entre 0 y 1, este umbral es para definir si un píxel es un borde o no, este umbral se establece de 0.5 por defecto, ya que es un punto medio, para así obtener resultados regulares, esto es de la siguiente forma: si el valor del píxel es mayor a 0.5, el píxel a la salida será igual a 1, y si el valor del píxel es menor a 0.5, el valor del píxel a la salida será igual a 0, con esto se manipulará, cuáles serán puntos de borde y cuáles no.

Antes de finalizar, hay que recordar que al principio de este capítulo, se mencionó, que si la imagen seleccionada no es múltiplo de 3, se hace múltiplo agregando "ceros" o "unos" al contorno de la imagen, y en caso contrario se deja tal cual.

En caso de que no sea múltiplo de 3, y se hayan agregado "ceros" o "unos", éstos se tendrán que eliminar del contorno de la imagen, para así tener el tamaño original de la imagen seleccionada.

Una vez finalizado todo esto, se puede observar el resultado final, obteniendo así los bordes de la imagen seleccionada, y así también se pueden observar resultados satisfactorios, muy similares a algunos métodos ya definidos, como son el de: Robert, Prewitt, Sobel, etc.

Capítulo 4 Pruebas y Resultados

En este capítulo, se pueden observar los resultados del método desarrollado en el capítulo 3, aquí se podrán observar los resultados previamente obtenidos de las pruebas hechas, así como el resultado final y cómo es que se llega a esta conclusión.

4.1 Eigenvectores a probar

Como ya se había descrito anteriormente, se descartaron los eigenvectores correspondientes a la matriz de mínimos, y se seleccionaron los eigenvectores 4, 6, 7 y 8, correspondientes a la matriz de máximos, que posteriormente fueron pasados a forma de matriz de 3x3, para aplicarse a la imagen seleccionada.

Con estas 4 matrices de 3x3, se pueden obtener diferentes combinaciones entre 2 máscaras, para así aplicar estas máscaras a la imagen seleccionada, las cuales son: (4 con 4), (4 con 6), (4 con 7), (4 con 8), (6 con 6), (6 con 7), (6 con 8), (7 con 7), (7 con 8) y (8 con 8).

Se utilizan dos máscaras, porque basándose en algoritmos ya establecidos, como son, el de Roberts, Sobel, Prewitt, etc., se puede notar que usan 2 máscaras: una para detectar bordes horizontales y la otra para detectar bordes verticales, también se puede notar que ambas máscaras son iguales, la diferencia es que la segunda es una variante de la primera.

Para ejemplificar esto, se tienen las siguientes máscaras, pertenecientes a Sobel (Fig. 4.1) y Prewitt (Fig. 4.2).

$$\Delta x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \Delta y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Fig. 4.1 Máscaras de Sobel

$$\Delta x = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad \Delta y = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Fig. 4.2 Máscaras de Prewitt

Como se puede observar, estas máscaras tienen una particularidad, que la segunda máscara, Δy , es la transpuesta de la primera máscara Δx , por esta razón, utilizamos dos máscaras para nuestro algoritmo.

Basándonos en esto, observando que la segunda máscara es la transpuesta de la primera, por consiguiente, se pueden descartar algunas combinaciones mencionadas anteriormente, quedando únicamente: (4 con 4), (6 con 6), (7 con 7) y (8 con 8).

Se quedaron estas combinaciones, porque recordando lo anteriormente mencionado, se decía que la segunda máscara es la transpuesta de la primera, quedando entonces así de la siguiente manera, las posibles combinaciones: (4 con 4'), (6 con 6'), (7 con 7') y (8 con 8')¹.

Estas últimas posibles combinaciones fueron las combinaciones finales, con las cuales se trabajará para la obtención del resultado.

Una vez obtenido esto, se procede a la tarea de aplicar estas máscaras a la imagen, para posteriormente sumar las imágenes con sus valores absolutos, como se explicó en el último paso del capítulo anterior.

Y con esto se pueden observar los resultados, para ver cuál eigenvector es el que mejor soluciona nuestro problema, para mostrar el resultado final.

4.2 Pruebas para seleccionar eigenvector

Como ya habíamos mencionado antes, vamos a hacer pruebas con los eigenvectores 4, 6, 7 y 8, para ver cuál es el que soluciona nuestro problema.

Prueba 1

La Fig. 4.3 representa la imagen original, para la prueba 1.

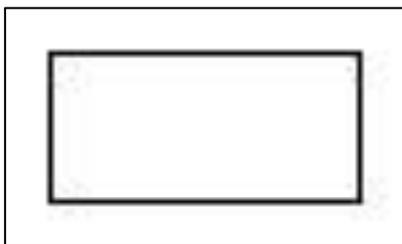


Fig. 4.3 Rectángulo de prueba

¹ "La comilla simple seguida del número quiere decir que es la transpuesta"

Capítulo 4

Resultados correspondientes al eigenvector 4, la Fig. 4.4 sirve para la detección de los bordes verticales, mientras que la Fig. 4.5, sirve para la detección de los bordes horizontales, como se muestra enseguida:



Fig. 4.4 Máscara 4 aplicada al rectángulo



Fig. 4.5 Máscara 4 transpuesta aplicada al rectángulo

La Fig. 4.6 muestra la suma de ambos resultados obtenidos, correspondientes al eigenvector 4, después de aplicar las máscaras 4 y 4 transpuesta a la Fig. 4.3, como se ve enseguida.



Fig. 4.6 Resultado del rectángulo con eigenvector 4

Resultados correspondientes al eigenvector 6, la Fig. 4.7 sirve para la detección de los bordes verticales, mientras que la Fig. 4.8, sirve para la detección de los bordes horizontales, como se muestra enseguida:

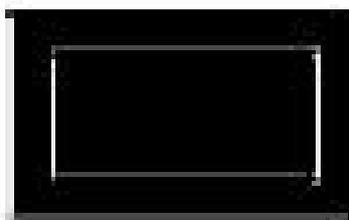


Fig. 4.7 Máscara 6 aplicada al rectángulo

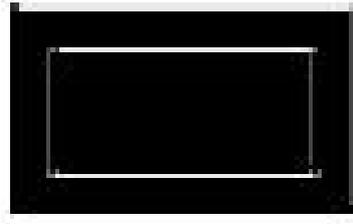


Fig. 4.8 Máscara 6 transpuesta aplicada al rectángulo

La Fig. 4.9 muestra la suma de ambos resultados obtenidos, correspondientes al eigenvector 6, después de aplicar las máscaras 6 y 6 transpuesta a la Fig. 4.3, como se ve enseguida.



Fig. 4.9 Resultado del rectángulo con eigenvector 6

Resultados correspondientes al eigenvector 7, la Fig. 4.10 sirve para la detección de los bordes verticales, mientras que la Fig. 4.11, sirve para la detección de los bordes horizontales, como se muestra enseguida:



Fig. 4.10 Máscara 7 aplicada al rectángulo



Fig. 4.11 Máscara 7 transpuesta aplicada al rectángulo

Capítulo 4

La Fig. 4.12 muestra la suma de ambos resultados obtenidos, correspondientes al eigenvector 7, después de aplicar las máscaras 7 y 7 transpuesta a la Fig. 4.3, como se ve enseguida.



Fig. 4.12 Resultado del rectángulo con eigenvector 7

Resultados correspondientes al eigenvector 8, la Fig. 4.13 sirve para la detección de los bordes verticales, mientras que la Fig. 4.14, sirve para la detección de los bordes horizontales, como se muestra enseguida:

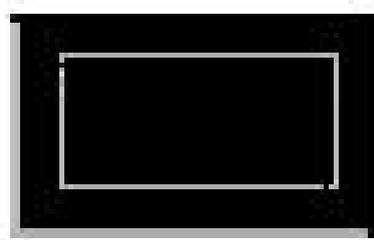


Fig. 4.13 Máscara 8 aplicada al rectángulo



Fig. 4.14 Máscara 8 transpuesta aplicada al rectángulo

La Fig. 4.15 muestra la suma de ambos resultados obtenidos, correspondientes al eigenvector 8, después de aplicar las máscaras 8 y 8 transpuesta a la Fig. 4.3, como se ve enseguida.



Fig. 4.15 Resultado del rectángulo con eigenvector 8

Prueba 2

La Fig. 4.16 representa la imagen original, para la prueba 2.

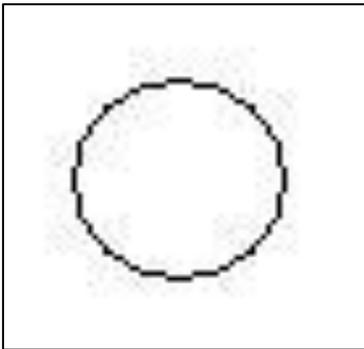


Fig. 4.16 Círculo de prueba

Resultados correspondientes al eigenvector 4, la Fig. 4.17 sirve para la detección de los bordes verticales, mientras que la Fig. 4.18, sirve para la detección de los bordes horizontales, como se muestra enseguida:

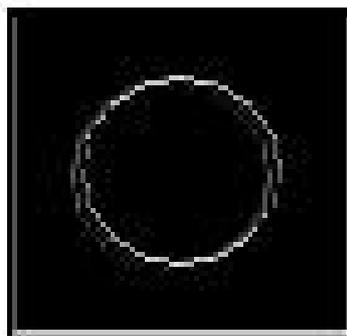


Fig. 4.17 Máscara 4 aplicada al círculo

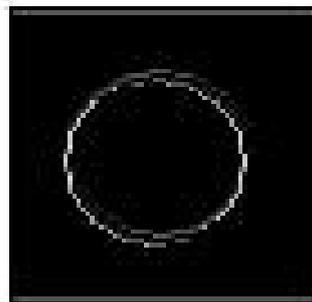


Fig. 4.18 Máscara 4 transpuesta aplicada al círculo

Capítulo 4

La Fig. 4.19 muestra la suma de ambos resultados obtenidos, correspondientes al eigenvector 4, después de aplicar las máscaras 4 y 4 transpuesta a la Fig. 4.16, como se ve enseguida.

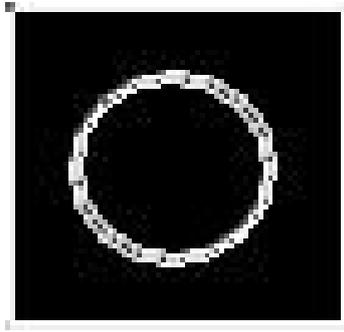


Fig. 4.19 Resultado del círculo con eigenvector 4

Resultados correspondientes al eigenvector 6, la Fig. 4.20 sirve para la detección de los bordes verticales, mientras que la Fig. 4.21, sirve para la detección de los bordes horizontales, como se muestra enseguida:

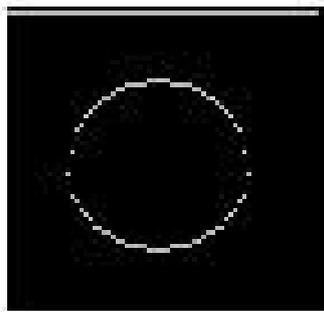


Fig. 4.20 Máscara 6 aplicada al círculo

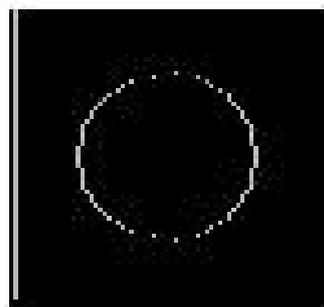


Fig. 4.21 Máscara 6 transpuesta aplicada al círculo

Capítulo 4

La Fig. 4.22 muestra la suma de ambos resultados obtenidos, correspondientes al eigenvector 6, después de aplicar las máscaras 6 y 6 transpuesta a la Fig. 4.16, como se ve enseguida.

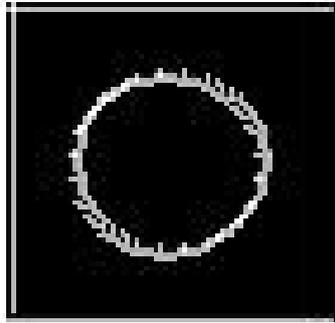


Fig. 4.22 Resultado del círculo con eigenvector 6

Resultados correspondientes al eigenvector 7, la Fig. 4.23 sirve para la detección de los bordes verticales, mientras que la Fig. 4.24, sirve para la detección de los bordes horizontales, como se muestra enseguida:

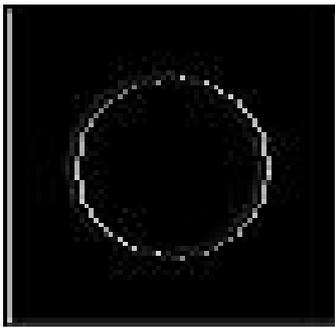


Fig. 4.23 Máscara 7 aplicada al círculo



Fig. 4.24 Máscara 7 transpuesta aplicada al círculo

Capítulo 4

La Fig. 4.25 muestra la suma de ambos resultados obtenidos, correspondientes al eigenvector 7, después de aplicar las máscaras 7 y 7 transpuesta a la Fig. 4.16, como se ve enseguida.

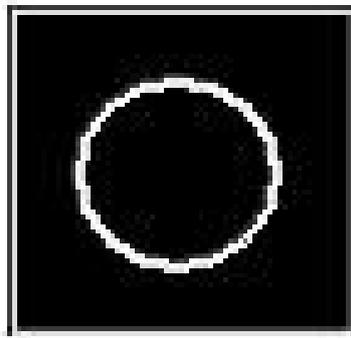


Fig. 4.25 Resultado del círculo con eigenvector 7

Resultados correspondientes al eigenvector 8, la Fig. 4.26 sirve para la detección de los bordes verticales, mientras que la Fig. 4.27, sirve para la detección de los bordes horizontales, como se muestra enseguida:



Fig. 4.26 Máscara 8 aplicada al círculo



Fig. 4.27 Máscara 8 transpuesta aplicada al círculo

Capítulo 4

La Fig. 4.28 muestra la suma de ambos resultados obtenidos, correspondientes al eigenvector 8, después de aplicar las máscaras 8 y 8 transpuesta a la Fig. 4.16, como se ve enseguida.

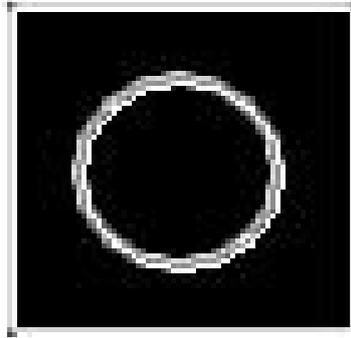


Fig. 4.28 Resultado del círculo con eigenvector 8

Prueba 3

La Fig. 4.29 representa la imagen original, para la prueba 3.



Fig. 4.29 Imagen de Lena 2

Capítulo 4

Resultados correspondientes al eigenvector 4, la Fig. 4.30 sirve para la detección de los bordes verticales, mientras que la Fig. 4.31, sirve para la detección de los bordes horizontales, como se muestra enseguida:



Fig. 4.30 Máscara 4 aplicada a la imagen de Lena 2



Fig. 4.31 Máscara 4 transpuesta aplicada a la imagen de Lena 2

Capítulo 4

La Fig. 4.32 muestra la suma de ambos resultados obtenidos, correspondientes al eigenvector 4, después de aplicar las máscaras 4 y 4 transpuesta a la Fig. 4.29, como se ve enseguida.



Fig. 4.32 Resultado de la imagen de Lena 2 con eigenvector 4

Resultados correspondientes al eigenvector 6, la Fig. 4.33 sirve para la detección de los bordes verticales, mientras que la Fig. 4.34, sirve para la detección de los bordes horizontales, como se muestra enseguida:



Fig. 4.33 Máscara 6 aplicada a la imagen de Lena 2



Fig. 4.34 Máscara 6 transpuesta aplicada a la imagen de Lena 2

La Fig. 4.35 muestra la suma de ambos resultados obtenidos, correspondientes al eigenvector 6, después de aplicar las máscaras 6 y 6 transpuesta a la Fig. 4.29, como se ve enseguida.



Fig. 4.35 Resultado de la imagen de Lena 2 con eigenvector 6

Capítulo 4

Resultados correspondientes al eigenvector 7, la Fig. 4.36 sirve para la detección de los bordes verticales, mientras que la Fig. 4.37, sirve para la detección de los bordes horizontales, como se muestra enseguida:



Fig. 4.36 Máscara 7 aplicada a la imagen de Lena 2



Fig. 4.37 Máscara 7 transpuesta aplicada a la imagen de Lena 2

Capítulo 4

La Fig. 4.38 muestra la suma de ambos resultados obtenidos, correspondientes al eigenvector 7, después de aplicar las máscaras 7 y 7 transpuesta a la Fig. 4.29, como se ve enseguida.



Fig. 4.38 Resultado de la imagen de Lena 2 con eigenvector 7

Resultados correspondientes al eigenvector 8, la Fig. 4.39 sirve para la detección de los bordes verticales, mientras que la Fig. 4.40, sirve para la detección de los bordes horizontales, como se muestra enseguida:



Fig. 4.39 Máscara 8 aplicada a la imagen de Lena 2



Fig. 4.40 Máscara 8 transpuesta aplicada a la imagen de Lena 2

La Fig. 4.41 muestra la suma de ambos resultados obtenidos, correspondientes al eigenvector 8, después de aplicar las máscaras 8 y 8 transpuesta a la Fig. 4.29, como se ve enseguida.



Fig. 4.41 Resultado de la imagen de Lena 2 con eigenvector 8

Prueba 4

De esta prueba en adelante, vamos a mostrar los resultados de ambas máscaras, sumadas, debido a que ya se ha notado el resultado, de cuando se aplican solas y cuando se aplican ambas.

La Fig. 4.29 representa la imagen original, para la prueba 4.



Fig. 4.42 Siluetas

En la Fig. 4.43, se presentan los resultados obtenidos de sumar ambas máscaras, tanto de los eigenvectores 4, 6, 7 y 8, esto con el fin de observar los resultados finales que es lo que nos interesa, y hacemos lo mismo para las pruebas posteriores.

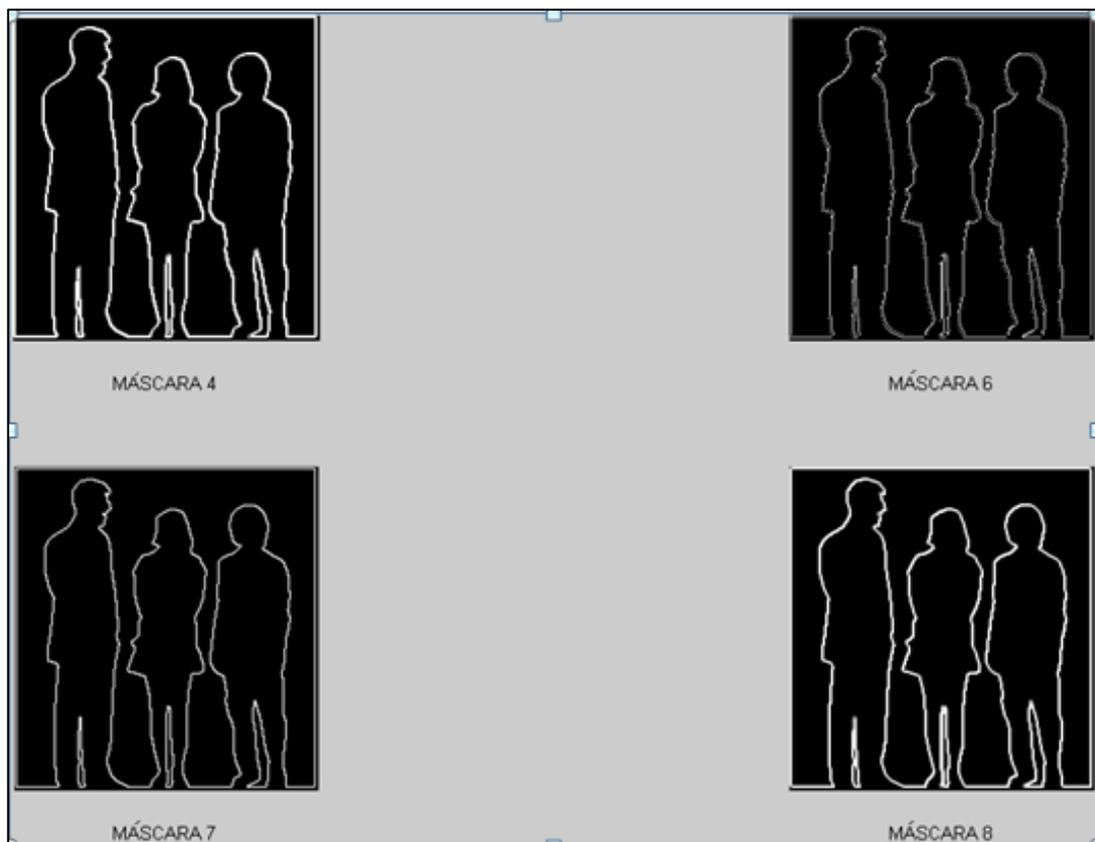


Fig. 4.43 Aplicación de distintas máscaras, a la imagen de siluetas

Prueba 5

La Fig. 4.44 representa la imagen original, para la prueba 5.



Fig. 4.44 Abeja

En la Fig. 4.45 se observan los resultados sumados para cada máscara seleccionada en este algoritmo, para su posterior evaluación.



Fig. 4.45 Aplicación de distintas máscaras, a la imagen de abeja

Prueba 6

La Fig. 4.46 representa la imagen original, para la prueba 6.



Fig. 4.46 Gato

En la Fig. 4.47 se observan los resultados sumados para cada máscara seleccionada en este algoritmo, para su posterior evaluación.

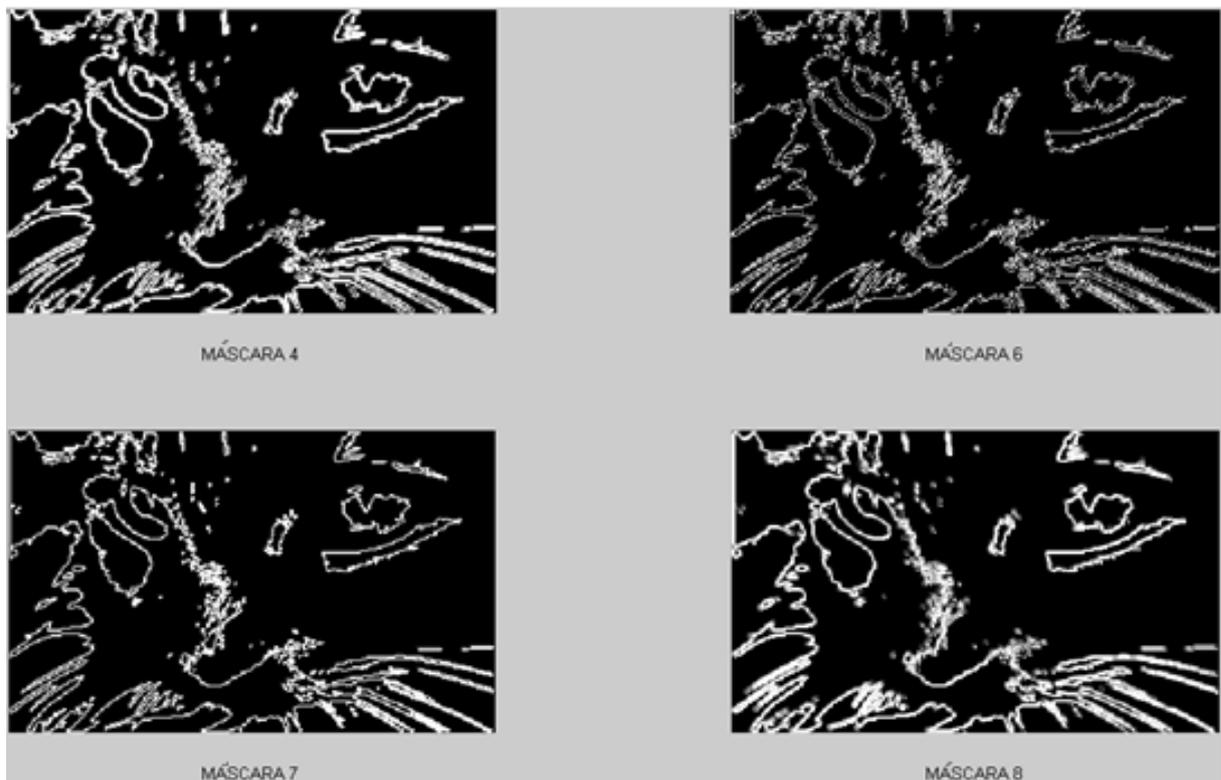


Fig. 4.47 Aplicación de distintas máscaras, a la imagen de gato

Prueba 7

La Fig. 4.48 representa la imagen original, para la prueba 7.



Fig. 4.48 Félix

En la Fig. 4.49 se observan los resultados sumados para cada máscara seleccionada en este algoritmo, para su posterior evaluación.

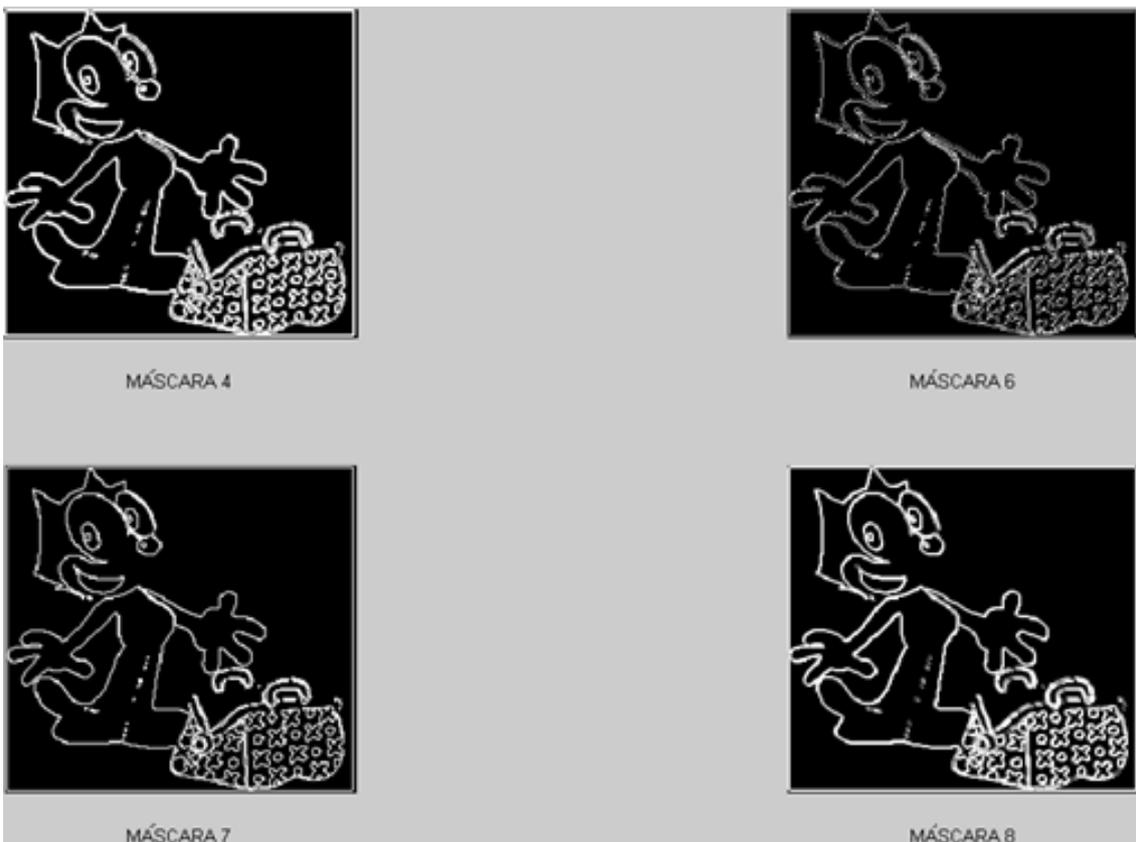


Fig. 4.49 Aplicación de distintas máscaras, a la imagen de Felix

En la figura 116, observamos más diferencias en las imágenes.

Prueba 8

La Fig. 4.50 representa la imagen original, para la prueba 8.



Fig. 4.50 Pingüino

En la Fig. 4.51 se observan los resultados sumados para cada máscara seleccionada en este algoritmo, para su posterior evaluación.



Fig. 4.51 Aplicación de distintas máscaras, a la imagen del pingüino

Capítulo 4

Con estos resultados obtenidos, haciendo un análisis cuidadoso, y comparando todas las imágenes, se concluye que los eigenvectores 7 y 8 son los que mejores resultados arrojan, pero el eigenvector 8 arroja los bordes un poco más gruesos, por lo que se concluye que la máscara que se utiliza, es la correspondiente a la matriz formada por el eigenvector 7, con este eigenvector se pueden observar los bordes de la imagen más definidos.

Es por esta razón, que en el capítulo 3, tomamos el eigenvector 7, para dar por finalizado este algoritmo.

4.3 Resultados finales

En esta sección, nos enfocamos a lo que son los resultados obtenidos con el eigenvector 7, aplicado como máscara a diferentes imágenes, para así verificar su respuesta.

En la Fig. 4.52, tenemos la imagen original.

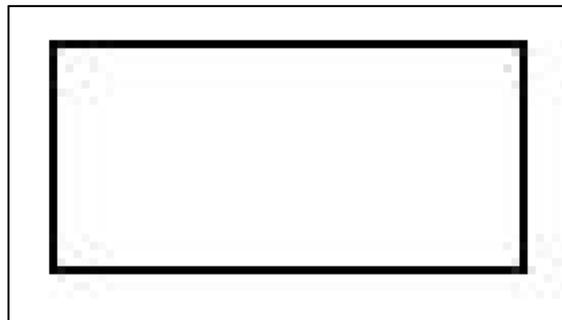


Fig. 4.52 Rectángulo

En la Fig. 4.53, tenemos el resultado de aplicar el eigenvector 7 como máscara para la detección de bordes.



Fig. 4.53 Resultado con el rectángulo

Capítulo 4

En la Fig. 4.54, tenemos la imagen original.

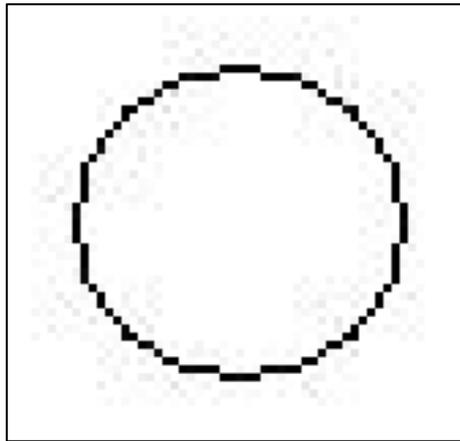


Fig. 4.54 Círculo

En la Fig. 4.55 tenemos el resultado de aplicar el eigenvector 7 como máscara para la detección de bordes.



Fig. 4.55 Resultado con el círculo

Capítulo 4

En la Fig. 4.56, tenemos la imagen original.



Fig. 4.56 Imagen de Lena 3

En la Fig. 4.57 tenemos el resultado de aplicar el eigenvector 7 como máscara para la detección de bordes.



Fig. 4.57 Resultado con la imagen de Lena 3

Capítulo 4

En la Fig. 4.58, tenemos la imagen original.



Fig 4. 58 Letra A

En la Fig. 4.59, tenemos el resultado de aplicar el eigenvector 7 como máscara para la detección de bordes.



Fig 4. 59 Resultado con la letra A

Capítulo 4

En la Fig. 4.60, tenemos la imagen original.

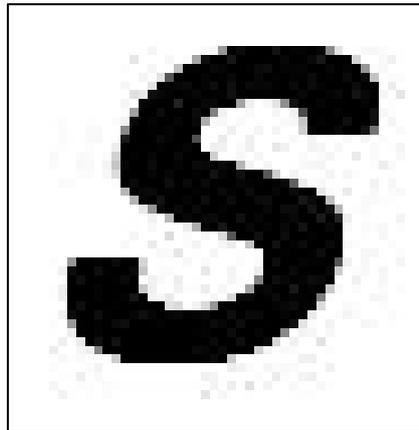


Fig 4. 60 Letra S

En la Fig. 4.61, tenemos el resultado de aplicar el eigenvector 7 como máscara para la detección de bordes.



Fig 4. 61 Resultado con la letra S

Capítulo 4

En la Fig. 4.62, tenemos la imagen original.

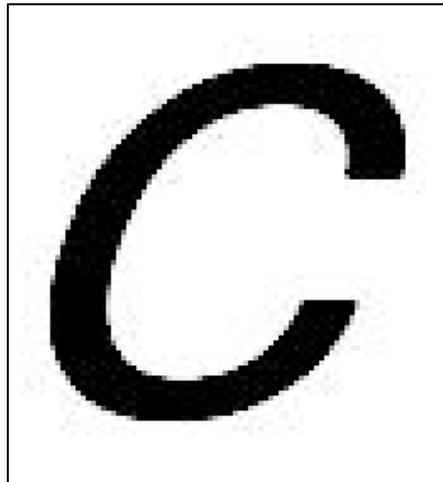


Fig 4. 62 Letra C

En la Fig. 4.63, tenemos el resultado de aplicar el eigenvector 7 como máscara para la detección de bordes.



Fig 4. 63 Resultado con la letra C

Capítulo 4

En la Fig. 4.64, tenemos la imagen original.



Fig 4. 64 Abeja 2

En la Fig. 4.65, tenemos el resultado de aplicar el eigenvector 7 como máscara para la detección de bordes.

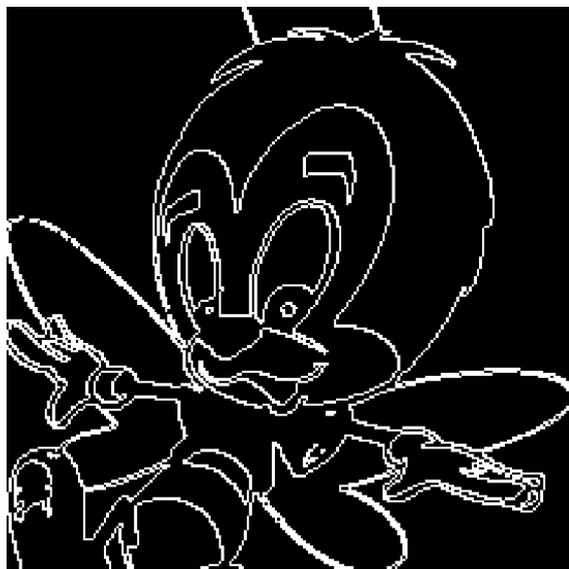


Fig 4. 65 Resultado con la abeja 2

Capítulo 4

En la Fig. 4.66, tenemos la imagen original.



Fig 4. 66 Siluetas 2

En la Fig. 4.67, tenemos el resultado de aplicar el eigenvector 7 como máscara para la detección de bordes.

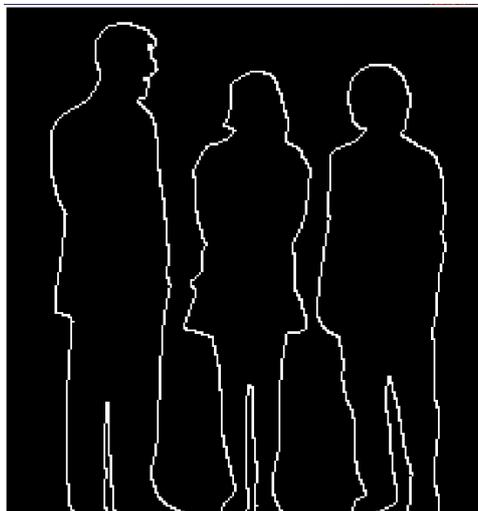


Fig 4. 67 Resultado con siluetas 2

Capítulo 4

En la Fig. 4.68, tenemos la imagen original.



Fig 4. 68 Félix 2

En la Fig. 4.69, tenemos el resultado de aplicar el eigenvector 7 como máscara para la detección de bordes.



Fig 4. 69 Resultado con Felix 2

Capítulo 4

En la Fig. 4.70, tenemos la imagen original.



Fig 4. 70 Escorpión

En la Fig. 4.71, tenemos el resultado de aplicar el eigenvector 7 como máscara para la detección de bordes.

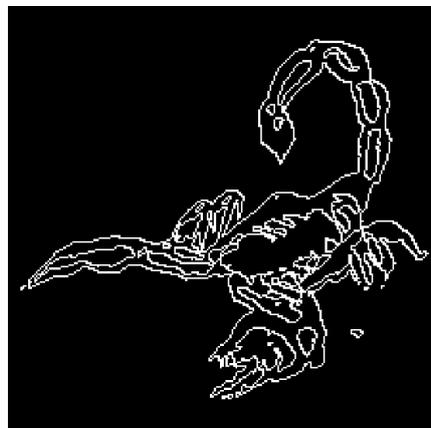


Fig 4. 71 Resultado con escorpión

Una vez observados todos los resultados, podemos dar por finalizado este algoritmo para la detección de bordes, mediante el uso de la memoria asociativa Alfa-Beta, obteniendo así resultados satisfactorios.

4.4 Comparación de resultados

En esta sección vamos a comparar nuestros resultados con los algoritmos ya existentes, como lo son: Robert, Sobel, Prewitt y Canny.

En la Fig.4.72 tenemos, la imagen original.



Fig. 4.72 Imagen seleccionada 1

Con el método de Prewitt tenemos la siguiente Fig. 4.73.



Fig. 4.73 Resultado 1 obtenido con el método de Prewitt

Con el método de Robert tenemos la Fig. 4.74.



Fig. 4.74 Resultado 1 obtenido con el método de Robert

Con el método de Sobel tenemos la Fig. 4.75.



Fig. 4.75 Resultado 1 obtenido con el método de Sobel

Con el método de Canny tenemos la Fig. 4.76.



Fig. 4.76 Resultado 1 obtenido con el método de Canny

Con nuestro método de propuesto, basado en las memorias asociativas Alfa-Beta, tenemos la Fig. 4.77.



Fig. 4.77 Resultado 1 obtenido con el método propuesto, utilizando las memorias asociativas Alfa-Beta

En la Fig.4.78 tenemos, la imagen original.



Fig. 4.78 Imagen seleccionada 2

Con el método de Prewitt tenemos la Fig. 4.79.

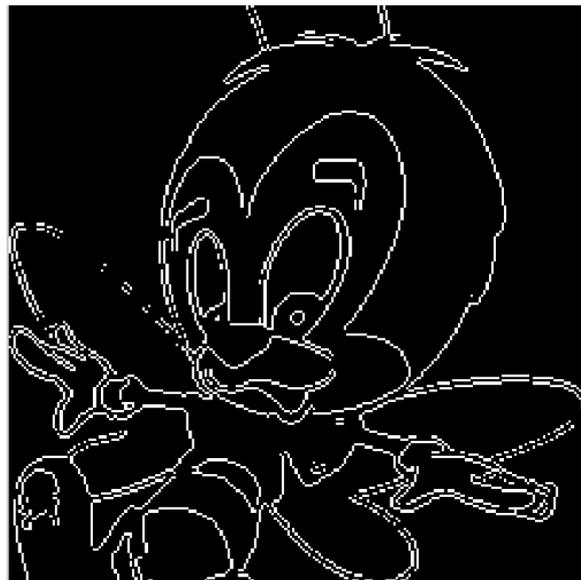


Fig. 4.79 Resultado 2 obtenido con el método de Prewitt

Con el método de Robert tenemos la Fig. 4.80.

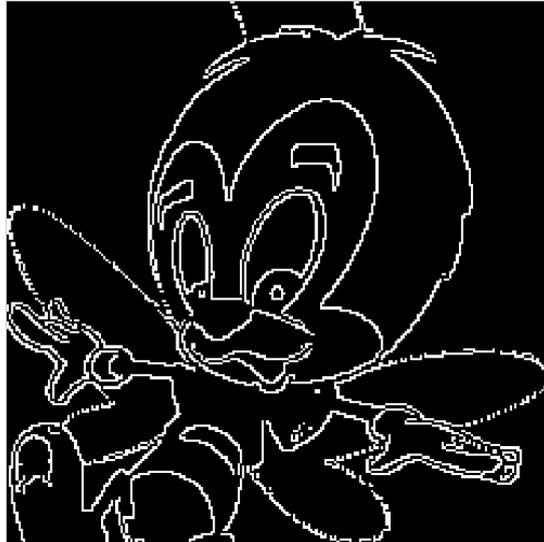


Fig. 4.80 Resultado 2 obtenido con el método de Robert

Con el método de Sobel tenemos la Fig. 4.81.

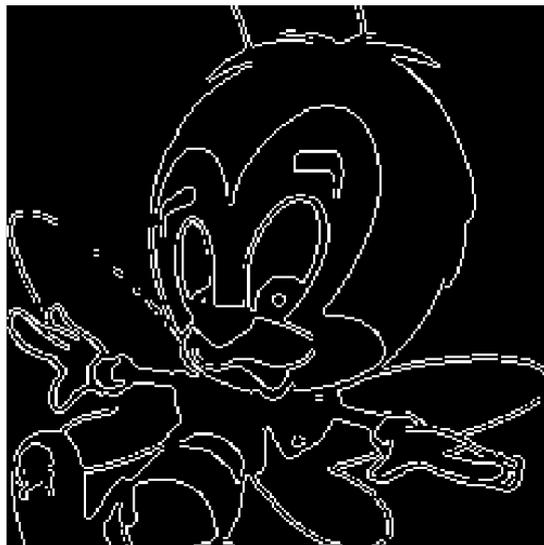


Fig. 4.81 Resultado 2 obtenido con el método de Sobel

Con el método de Canny tenemos la Fig. 4.82.

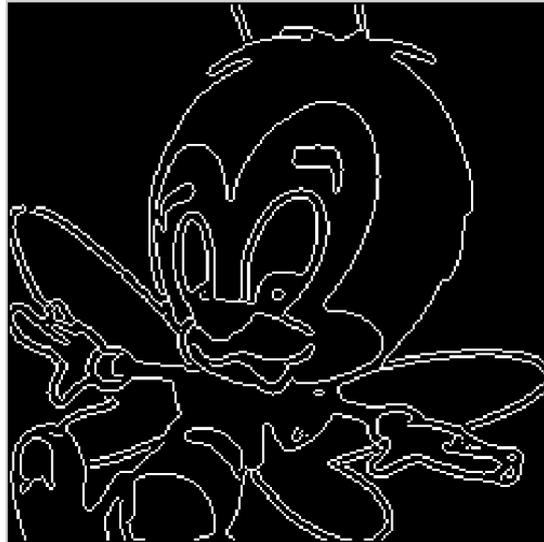


Fig. 4.82 Resultado 2 obtenido con el método de Canny

Con nuestro método de propuesto, basado en las memorias asociativas Alfa-Beta tenemos la Fig. 4.83.

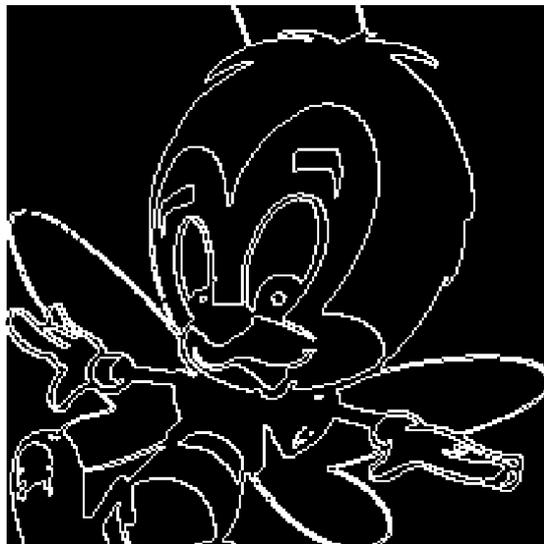


Fig. 4.83 Resultado 2 obtenido con el método propuesto, utilizando las memorias asociativas Alfa-Beta

Capítulo 4

En la Fig.4.84 tenemos, la imagen original.

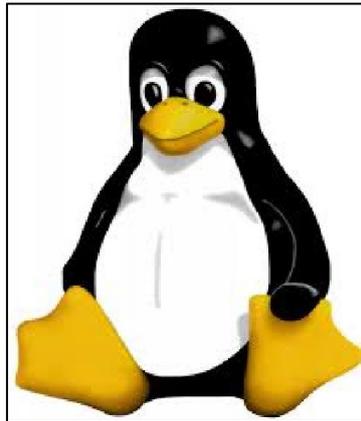


Fig. 4.84 Imagen seleccionada 3

Con el método de Prewitt tenemos la Fig. 4.85.

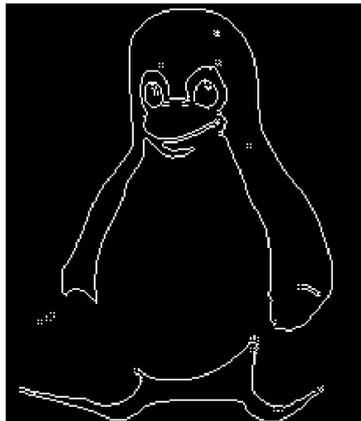


Fig. 4.85 Resultado 3 obtenido con el método de Prewitt

Con el método de Robert tenemos la Fig. 4.86.

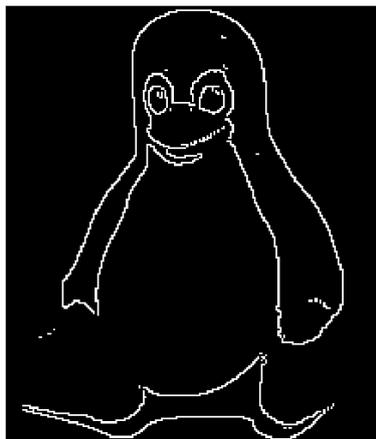


Fig. 4.86 Resultado 3 obtenido con el método de Robert

Capítulo 4

Con el método de Sobel tenemos la Fig. 4.87.

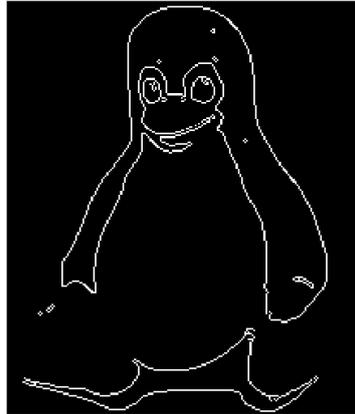


Fig. 4.87 Resultado 3 obtenido con el método de Sobel

Con el método de Canny tenemos la Fig. 4.88.

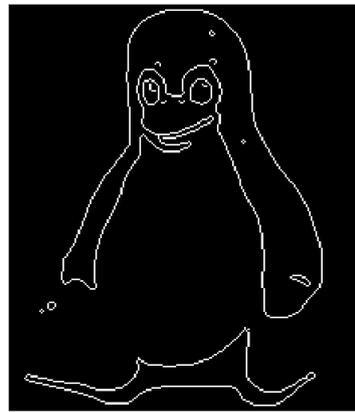


Fig. 4.88 Resultado 3 obtenido con el método de Canny

Con nuestro método de propuesto, basado en las memorias asociativas Alfa-Beta tenemos la Fig. 4.89.

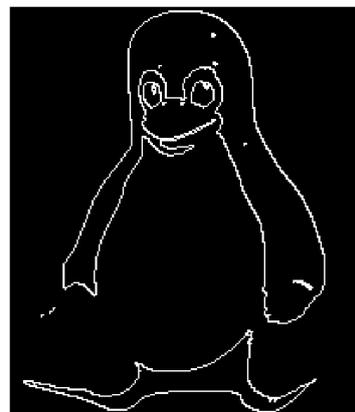


Fig. 4.89 Resultado 3 obtenido con el método propuesto, utilizando las memorias asociativas Alfa-Beta

Capítulo 4

En la Fig.4.90 tenemos, la imagen original.



Fig. 4.90 Imagen seleccionada 4

Con el método de Prewitt tenemos la Fig. 4.91.

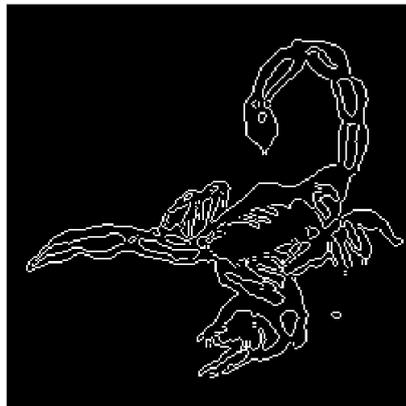


Fig. 4.91 Resultado 4 obtenido con el método de Prewitt

Con el método de Robert tenemos la Fig. 4.92.

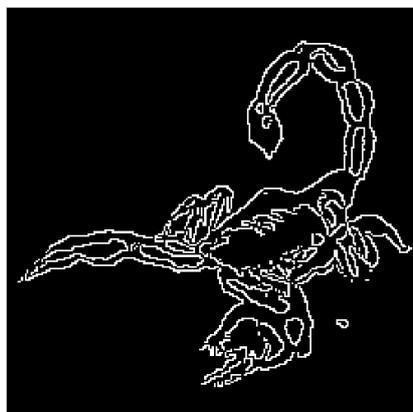


Fig. 4.92 Resultado 4 obtenido con el método de Robert

Con el método de Sobel tenemos la Fig. 4.93.

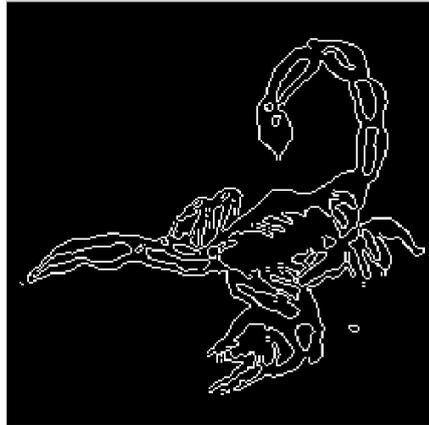


Fig. 4.93 Resultado 4 obtenido con el método de Sobel

Con el método de Canny tenemos la Fig. 4.94.

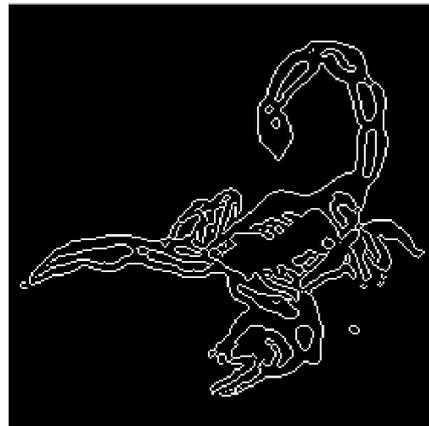


Fig. 4.94 Resultado 4 obtenido con el método de Canny

Con nuestro método de propuesto, basado en las memorias asociativas Alfa-Beta tenemos la Fig. 4.95.

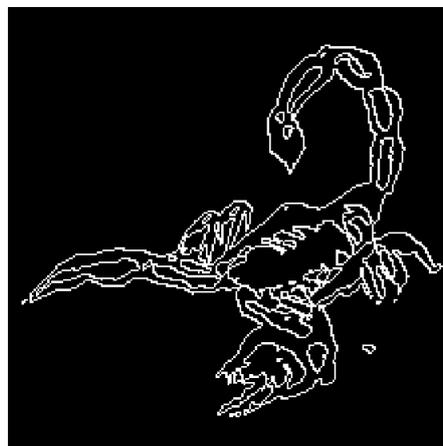


Fig. 4.95 Resultado 4 obtenido con el método propuesto, utilizando las memorias asociativas Alfa-Beta

En la Fig.4.96 tenemos, la imagen original.



Fig. 4.96 Imagen seleccionada 5

Con el método de Prewitt tenemos la Fig. 4.97.

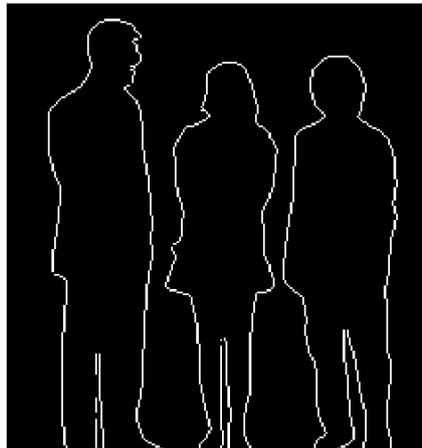


Fig. 4.97 Resultado 5 obtenido con el método de Prewitt

Con el método de Robert tenemos la Fig. 4.98.

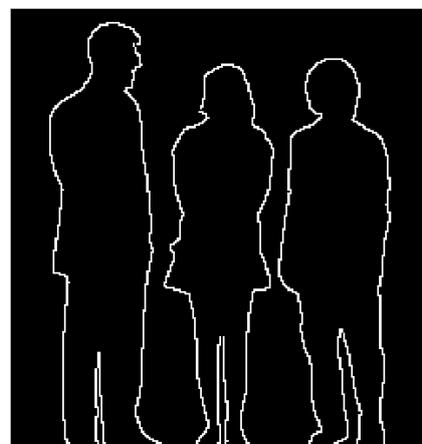


Fig. 4.98 Resultado 5 obtenido con el método de Robert

Con el método de Sobel tenemos la Fig. 4.99.

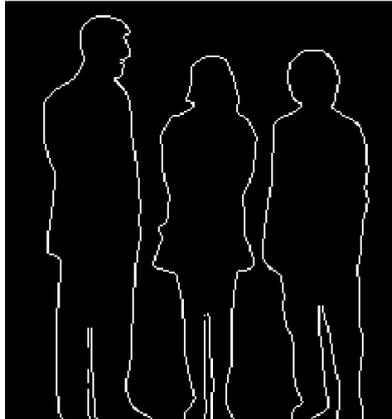


Fig. 4.99 Resultado 5 obtenido con el método de Sobel

Con el método de Canny tenemos la Fig. 4.100.

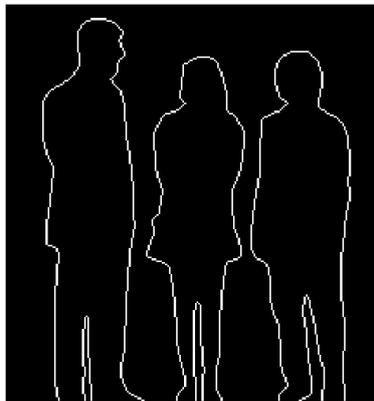


Fig. 4.100 Resultado 5 obtenido con el método de Canny

Con nuestro método de propuesto, basado en las memorias asociativas Alfa-Beta tenemos la Fig. 4.101.



Fig. 4.101 Resultado 5 obtenido con el método propuesto, utilizando las memorias asociativas Alfa-Beta

En la Fig.4.102 tenemos, la imagen original.



Fig. 4.102 Imagen seleccionada 6

Con el método de Prewitt tenemos la Fig. 4.103.

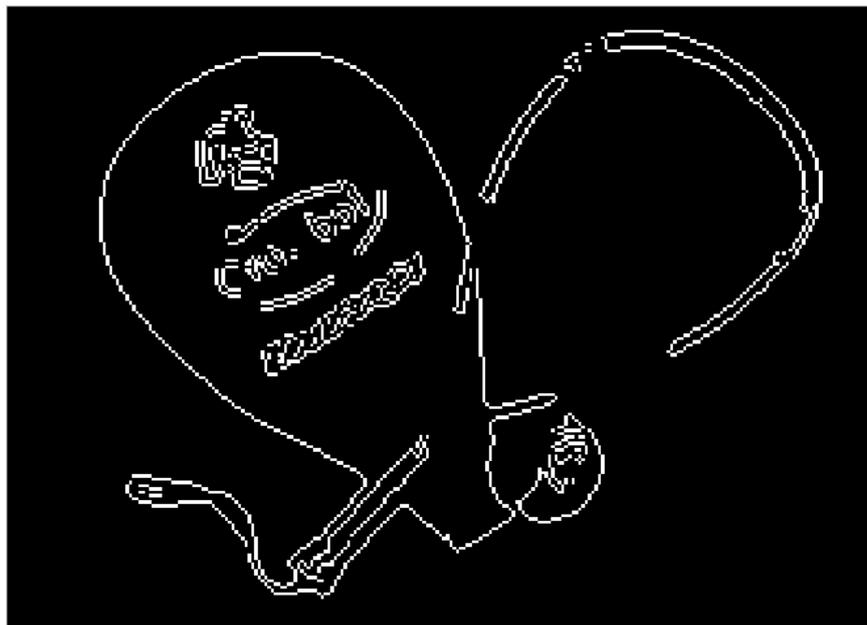


Fig. 4.103 Resultado 6 obtenido con el método de Prewitt

Con el método de Robert tenemos la Fig. 4.104.

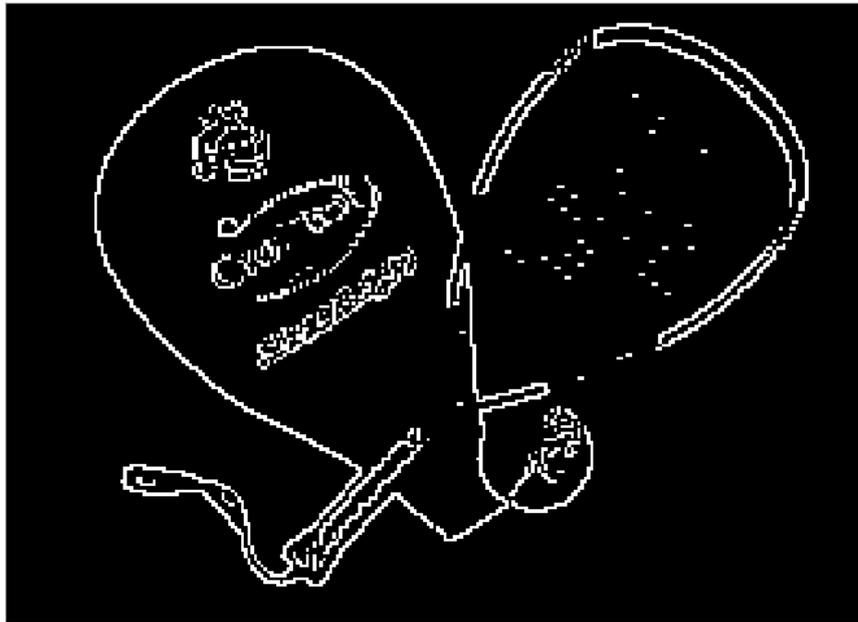


Fig. 4.104 Resultado 6 obtenido con el método de Robert

Con el método de Sobel tenemos la Fig. 4.105.

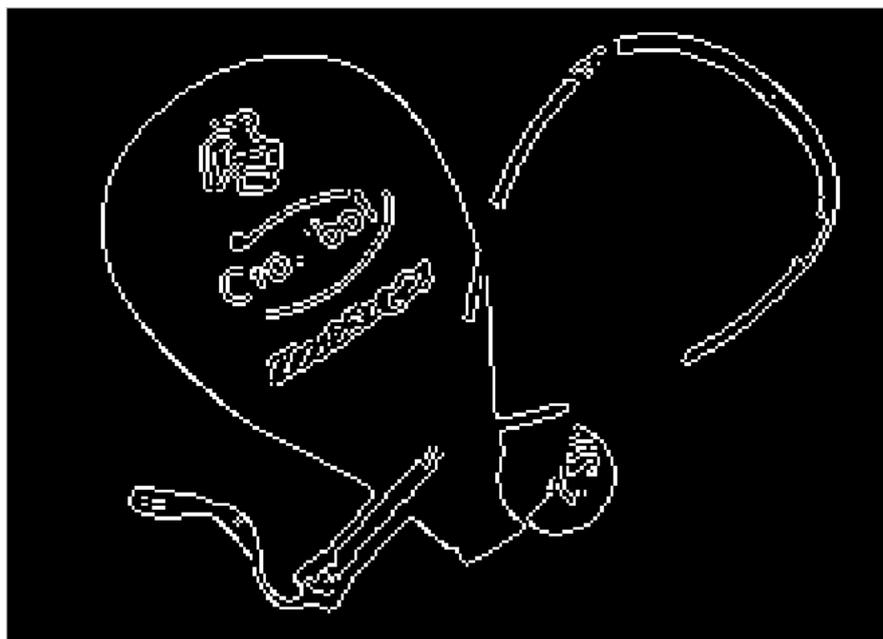


Fig. 4.105 Resultado 6 obtenido con el método de Sobel

Con el método de Canny tenemos la Fig. 4.106.

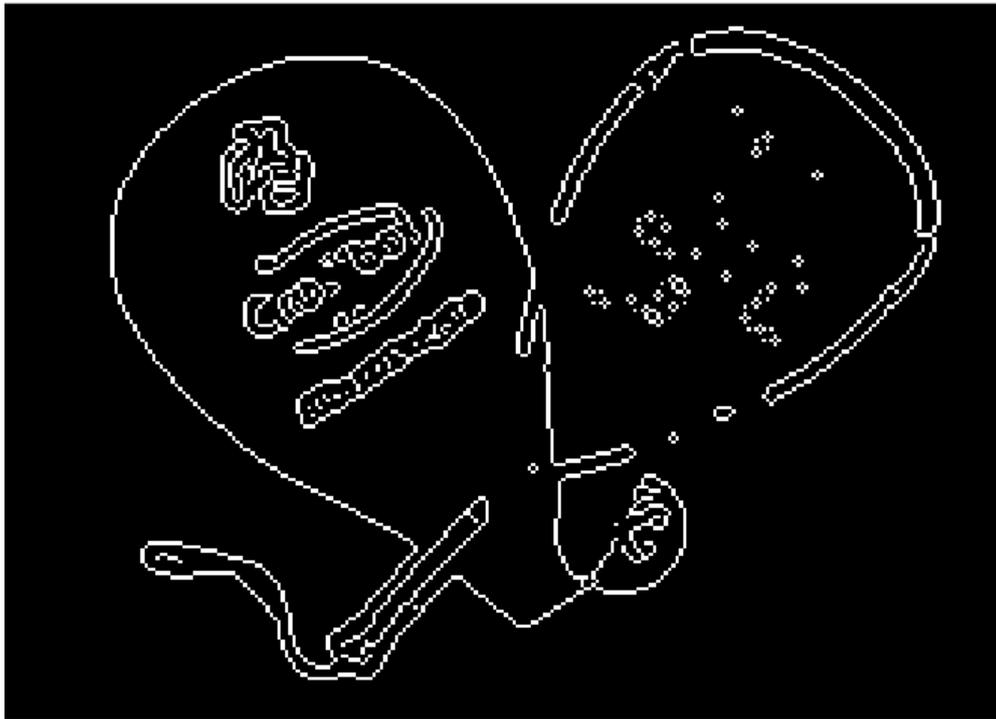


Fig. 4.106 Resultado 6 obtenido con el método de Canny

Con nuestro método de propuesto, basado en las memorias asociativas Alfa-Beta tenemos la Fig. 4.107.

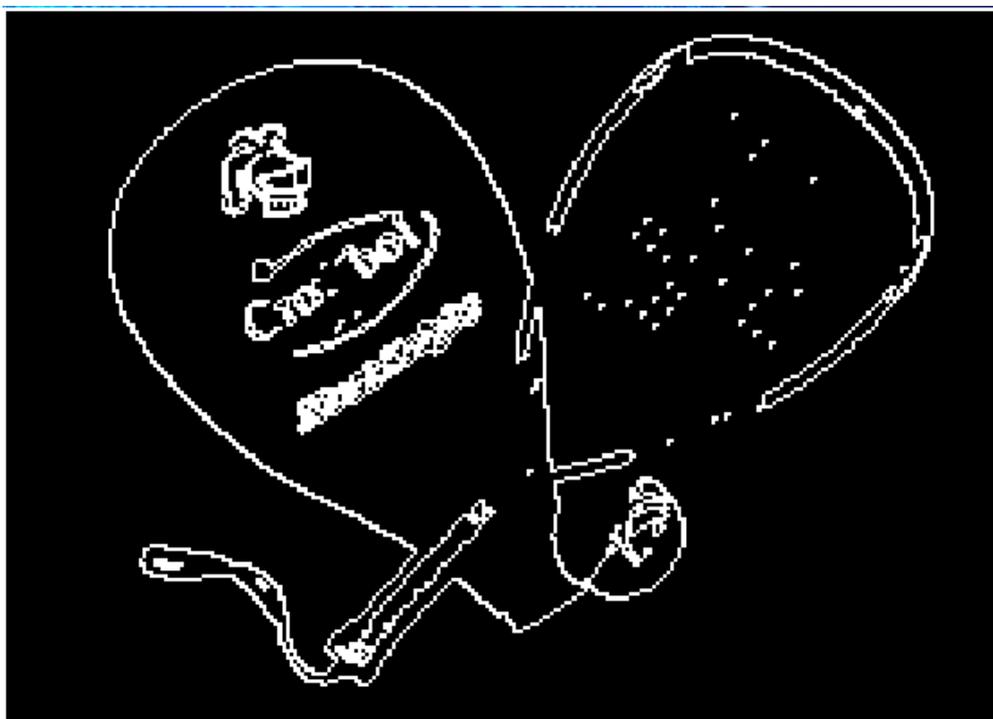


Fig. 4.107 Resultado 6 obtenido con el método propuesto, utilizando las memorias asociativas Alfa-Beta

Capítulo 4

En la Fig.4.108 tenemos, la imagen original.



Fig. 4.108 Imagen seleccionada 7

Con el método de Prewitt tenemos la Fig. 4.109.



Fig. 4.109 Resultado 7 obtenido con el método de Prewitt

Con el método de Robert tenemos la Fig. 4.110.

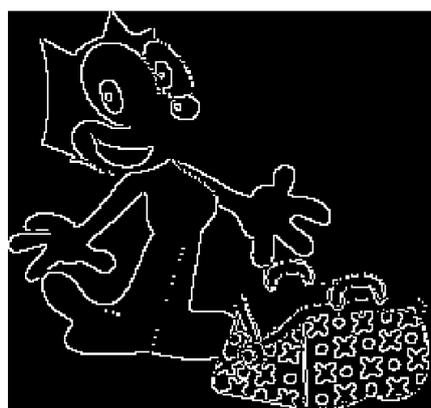


Fig. 4.110 Resultado 7 obtenido con el método de Robert

Con el método de Sobel tenemos la Fig. 4.111.



Fig. 4.111 Resultado 7 obtenido con el método de Sobel

Con el método de Canny tenemos la Fig. 4.112.



Fig. 4.112 Resultado 7 obtenido con el método de Canny

Con nuestro método de propuesto, basado en las memorias asociativas Alfa-Beta tenemos la Fig. 4.113.



Fig. 4.113 Resultado 7 obtenido con el método propuesto, utilizando las memorias asociativas Alfa-Beta

Capítulo 4

En la Fig.4.114 tenemos, la imagen original.



Fig. 4.114 Imagen seleccionada 8

Con el método de Prewitt tenemos la Fig. 4.115.

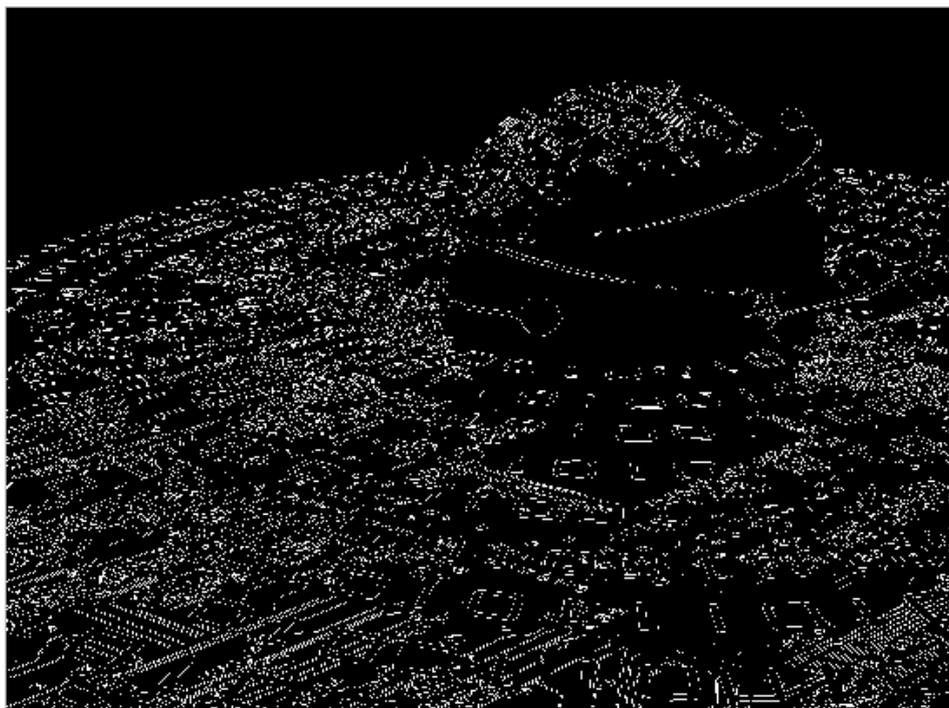


Fig. 4.115 Resultado 8 obtenido con el método de Prewitt

Con el método de Robert tenemos la Fig. 4.116.

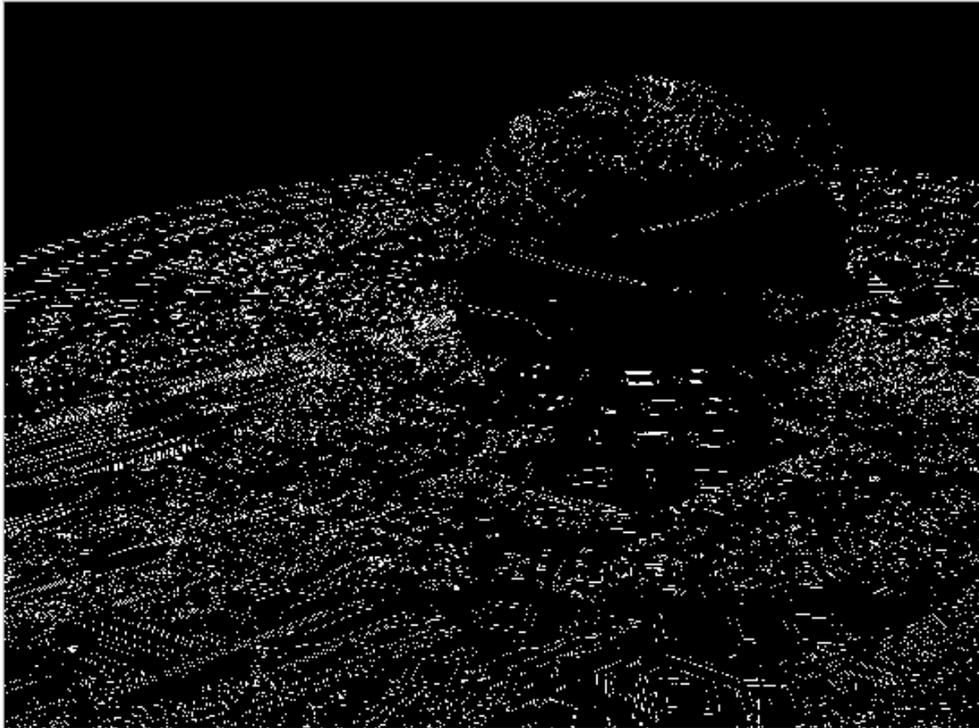


Fig. 4.116 Resultado 8 obtenido con el método de Robert

Con el método de Sobel tenemos la Fig. 4.117.

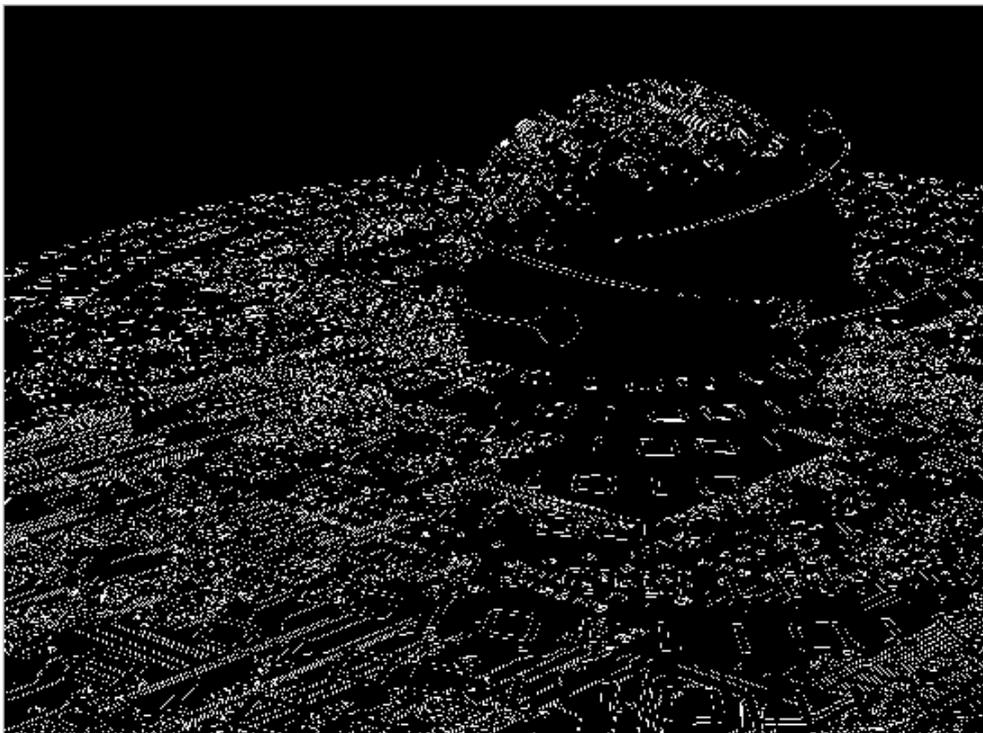


Fig. 4.117 Resultado 8 obtenido con el método de Sobel

Con el método de Canny tenemos la Fig. 4.118.

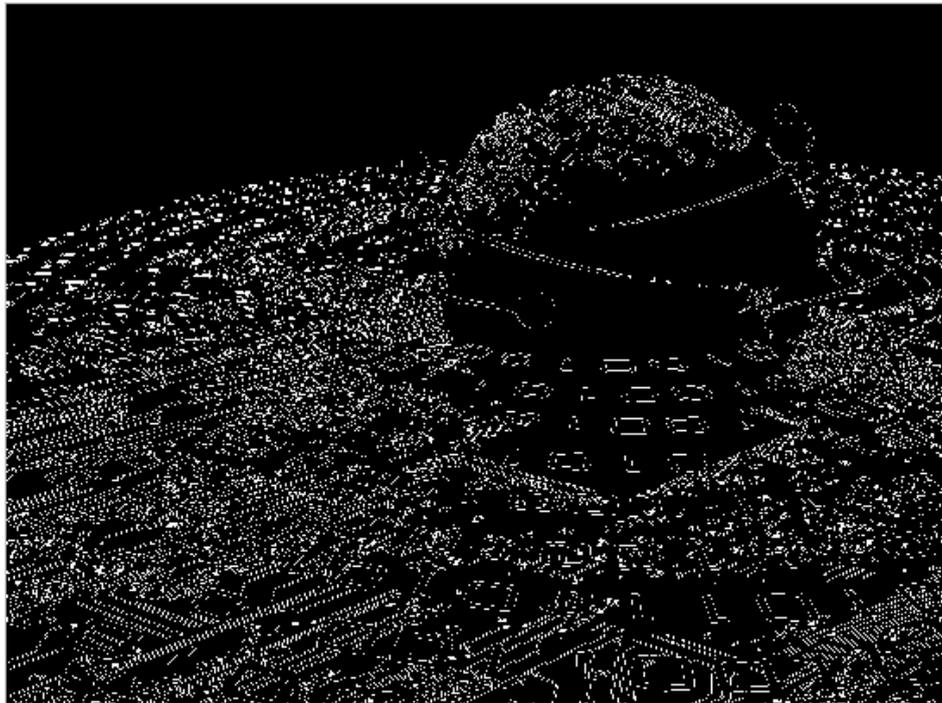


Fig. 4.118 Resultado 8 obtenido con el método de Canny

Con nuestro método de propuesto, basado en las memorias asociativas Alfa-Beta tenemos la Fig. 4.119.

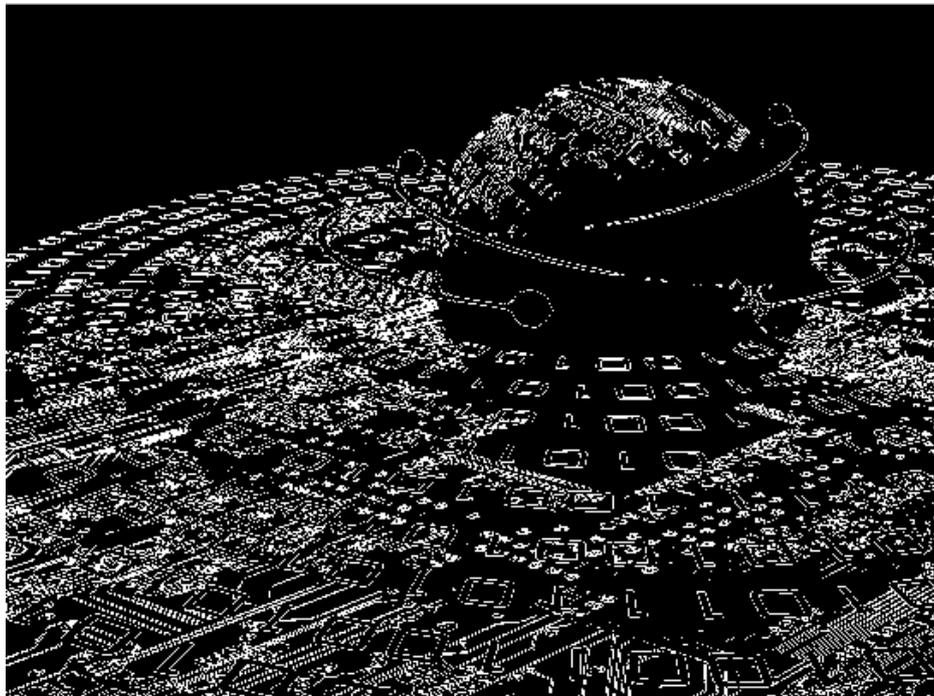


Fig. 4.119 Resultado 8 obtenido con el método propuesto, utilizando las memorias asociativas Alfa-Beta

Conclusiones

En el presente trabajo, se formuló un nuevo método para la detección de bordes, aunque ya hay diversos métodos para la detección de bordes en imágenes, este nuevo método apoyado en las memorias asociativas Alfa-Beta, resultó ser igual de eficaz que los métodos ya existentes, abriendo nuevas posibilidades, al enfoque macro de la inteligencia artificial, que ha estado un poco olvidado estos últimos años, ya que el enfoque micro que es donde se encuentran las redes neuronales, es el que ha tenido más avances últimamente, con esto podemos demostrar que las memorias asociativas, aún pueden llegar a tener un uso más amplio.

Gracias al estudio de las memorias asociativas, se logró efectuar este nuevo método, para detección de bordes, logrando implementar el algoritmo de las memorias alfa-beta.

Este nuevo algoritmo hecho con memorias asociativas, al llegar a ser efectivo, para la detección de bordes, así como los métodos anteriormente mencionados, y con el amplio rango de aplicaciones en áreas, tales como: identificación de patrones y control inteligente, brinda nuevas posibilidades para el campo de la inteligencia artificial.

En comparación con otros algoritmos para detección de bordes, por ejemplo: el de Robert, Sobel o Prewitt, podemos observar resultados muy similares, obteniendo una gran satisfacción, ya que se logró la meta de obtener un resultado satisfactorio.

Uno de los inconvenientes que surgieron a lo largo del desarrollo, fue que al obtener los eigenvectores con la ayuda del software MATLAB, los resultados obtenidos, fueron eigenvectores de forma compleja, es decir, con "parte real" y "parte imaginaria", por lo que se dio a la tarea de eliminar la parte compleja, quedándonos únicamente con la parte real, ya que la parte imaginaria era causante de malos resultados en la imagen de salida, es decir, se dieron valores no deseados y por consiguiente, no arrojaba resultados satisfactorios, además de que las imágenes no pueden llegar a tener valores complejos.

Fuera de esto, no hubo más inconvenientes en cuanto a este método, por lo que resultó ser un método no tan laborioso y con buenos resultados, para la detección de bordes, los resultados varían dependiendo de la imagen, y dependiendo de lo que se requiera, es aplicable este método, por ejemplo, se puede ver que en la Fig. 4.119, arrojó mejores resultados que los métodos con los cuales se estuvo comparando, pero en otras imágenes los resultados variaron, pero en general resultó ser un método alternativo más, para la detección de bordes.

Referencias

[1] Sitio Web:

www.tecnicaenlaboratorios.com/Nikon/Info_deteccion_de_bordes.htm

[2] Visión Computacional

L. Enrique Sucar Instituto Nacional de Astrofísica, Óptica y Electrónica

Puebla, México ----- Giovanni Gómez Helmholtz Zentrum Munchen

Neuherberg, Alemania

[3] Sitio Web:

http://www.varpa.org/~mgpenedo/cursos/Ip/Tema7/nodo7_1.html

[4] Trabajo citado:

Fundamentos de visión por computadora - sistemas informáticos avanzados
universitat JAUME.I

[5] F. Escolano, O. Colomina, M.A.Cazorla. Visión Artificial: Extracción de
Características I, 2006.

[6] Hill Green. Canny Edge Detection Tutorial. 2002.

[7] Trabajo citado:

Detección de bordes mediante algoritmo de Canny.

Autor: Jorge Valverde Rebaza

Escuela académico profesional de informática

Universidad nacional de Trujillo

[8] Mecánica Computacional Vol XXX, págs. 2841-2852 (artículo completo)
Oscar Möller, Javier W. Signorelli, Mario A. Storti (Eds.)
Rosario, Argentina, 1-4 Noviembre 2011
Ledda I. Larcher, Enrique M. Biasoni, Carlos A. Cattaneo, Ana I. Ruggeri,
A.Cecilia Herrera

[9] Algoritmo de bordes activos (active contours - snakes).

Autores:

Felipe Ramón Fabresse.

Juan Francisco Lerma Sánchez.

Jesús Rodríguez Hortal.

Año de Publicación (2010-2011).

[10] Detección de placas de matrícula y su posterior emborronado.

Autores:

Carlos Piñar Hafner.

Juan Manuel Gutiérrez Adanez.

Pablo Harillo Estanislao.

Año de Publicación (2009-2010).

[11] Un nuevo algoritmo de detección de bordes en imágenes con ruido gaussiano.

Autores:

Javier Fernando Vargas Caro.

Mohammed Gharbi.

Alejandro Lucas Zubillaga.

Año de publicación (2010-2011).

[12] Método de marcas de agua dual.

Autores: José Manuel Camacho Sosa.

José Antonio Pozo Núñez.

Gabriel Enrique Muñoz Ríos.

Año de Publicación (2012-2013).

[13] Reconocimiento de imágenes mediante redes neuronales.

Autores:

Domingo María Llorente Rivera.

Israel Camacho Ruano.

Nicolás Bellocchio.

Año de Publicación (2011-2012).

[14] Método para eliminar ruido impulsivo en imágenes a color usando técnicas fuzzy.

Autores:

Rafael Marín Nieto.

José Andrés García Romero De La Osa.

Daniel Pavón Pérez

Año de Presentación (2010-2011).

[15] Algoritmo para reconocimiento de patrones y búsqueda de imágenes.

Autores:

Pablo Aldana Caballero.

Adrián Díaz Herrero.

David Harillo Sánchez.

Darío Veledo García.

Año de Presentación (2009-2010).

[16] Algebra lineal

Sexta edición

Autor: Stanley I. Grossman

Editorial: Mc Graw Hill

[17] Sitio Web:

<http://jc-info.blogspot.mx/2011/06/detector-bordes-sobel-codigo-matlab.html>

[18] Sitio Web:

<http://www.mathworks.com/>

[19] Visión por computador

Segunda edición

Gonzalo Pajares Martinsanz

Jesús M. de la Cruz García

Alfaomega RA-MA

[20] VII Jornadas de Matemática Aplicada

DMA-IMPA

Universidad Politécnica de Valencia

22-23 de Noviembre de 2006

Páginas 9-20

Detección de bordes con precisión subpíxel en Imágenes digitales:
Interpolación lineal frente a esquemas de tipo no lineal.

[21] Trabajo citado:

Tesis doctoral.

Título.-Contribución al reconocimiento de objetos 2D mediante detección de bordes en imágenes a color.

Autor.-Nicolás Luis Fernández García

Lugar y Año de Publicación: Córdoba mayo 2002

[22] Trabajo citado:

Detección de bordes en una imagen.

Lugar.- Universidad de JAEN

Departamento de Ingeniería electrónica, Telecomunicación y Automática.

Área de Ingeniería de Sistemas y Automática

[23] Trabajo citado:

Tesis Bordes en Imágenes.

Autor: Juan Pablo Paredes Muñoz

Lugar de Publicación ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO
FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

[24] Trabajo citado:

Tesis.

ESCUELA DE INGENIERÍA ELECTRÓNICA EN TELECOMUNICACIONES Y REDES.

Autor: Leidy Tatiana Guerrero García

Lugar y Año de Publicación Riobamba – Ecuador 2012

[25] Trabajo citado:

Tesis.

Clasificación de patrones con memorias asociativas bidireccionales.

Autores: Sandra Paola Torres Jurado y Jesús Díaz García

[26] Sitio Web:

<http://oefa.blogspot.mx/2009/04/deteccion-de-bordes-algoritmo-de-canny.html>

[27] Sitio Web:

Procesamiento de Imágenes Digitales 2012-2013:

<https://opera-portal.us.es/pid/public/default/trabajo>

[28] Visión por computador

© 2003 - José Francisco Vélez Serrano, Ana Belén Moreno Díaz, Ángel Sánchez Calle, José Luis Esteban Sánchez-Marín

Apéndice A

Introducción

En este apartado, se definirán los valores y vectores característicos de una matriz cuadrada, por lo que una dimensión puede representarse por medio de matrices, lo que se trata de encontrar, son ecuaciones relacionadas con un sistema gráfico.

Sea $T:V \rightarrow W$ una transformación lineal. En diversas aplicaciones (una de las cuales se da en la siguiente sección), resulta útil encontrar un vector v en V , tal que Tv y v son paralelos. Es decir, se busca un vector v y un escalar λ , tal que [16]:

$$Tv = \lambda v \quad (28)$$

Si $v \neq 0$ y λ satisface (28), entonces λ se denomina un **valor característico** de T y v un **vector característico** de T , correspondiente al valor característico λ . El propósito de esta sección es, investigar las propiedades de los valores característicos y vectores característicos. Si V tiene dimensión finita, entonces T se puede representar por una matriz A_T . Por esta razón, se estudiarán los valores característicos de las matrices de $n \times n$.

Definición 1

Valor característico y vector característico

Sea A una matriz de $n \times n$ con componentes reales. El número λ (real o complejo), se denomina **valor característico** de A , si existe un vector **diferente de cero** v en \mathbb{C}^n , tal que:

$$Av = \lambda v \quad (29)$$

El vector $v \neq 0$, se denomina **vector característico** de A correspondiente al **valor característico** λ .

Nota. Los valores y vectores característicos, también se denominan **valores y vectores propios** o **eigenvalores y eigenvectores**: la palabra "eigen" es una palabra alemana para "propio".

Ejemplo 1

$$\text{Sea } A = \begin{pmatrix} 10 & -18 \\ 6 & -11 \end{pmatrix} \text{ Entonces } A \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 10 & -18 \\ 6 & -11 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

Así $\lambda_1 = 1$ es un valor característico de A con el correspondiente vector característico $v_1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$

De manera similar $A \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 10 & -18 \\ 6 & -11 \end{pmatrix} \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} -6 \\ -4 \end{pmatrix} = -2 \begin{pmatrix} 3 \\ 2 \end{pmatrix}$ = de modo que $\lambda_2 = -2$ es un valor característico de A con el correspondiente vector característico $v_2 = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$

Como se verá enseguida, éstos son los únicos valores característicos de A .

Ejemplo 2

Valores característicos y vectores característicos de la matriz identidad.

Sea $A = I$, entonces para cualquier $v \in \mathbb{C}^n$, $Av = Iv = v$ así, 1 es el único valor característico de A y todo $v \neq 0 \in \mathbb{C}^n$ es un vector característico de I .

Se calcularán los valores y vectores característicos de múltiples matrices en esta sección. Pero primero es necesario probar algunas técnicas que simplificarán estos cálculos.

Suponga que λ es un valor característico de A . Entonces existe un vector diferente de cero.

$v = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \neq 0$ Tal que $Av = \lambda v = \lambda / v$. Reescribiendo esto, se obtiene la

ecuación (30):

$$(A - \lambda I) = 0. \quad (30)$$

Si A es una matriz de $n \times n$, la ecuación (30) corresponde a un sistema homogéneo de n ecuaciones, con las incógnitas x_1, x_2, \dots, x_n .

Como se ha supuesto que el sistema cuenta con soluciones no triviales, se concluye de $\det(A - \lambda I) = 0$.

De forma inversa, si $\det(A - \lambda I) = 0$, entonces la ecuación (30) tiene soluciones no triviales y λ es el valor característico de A .

Por otro lado, si $\det(A - \lambda I) \neq 0$, entonces la única solución a (30) es $v = 0$, de manera que λ **no** es un valor característico de A . Resumiendo estos hechos, se tiene el siguiente teorema:

Teorema 1.

Sea A una matriz de $n \times n$. Entonces λ es un valor característico de A sí y sólo si

$$p(\lambda) = \det(A - \lambda I) = 0$$

Sea una matriz cuadrada, si le aplicamos la fórmula $(A - \lambda I)x = 0$, posteriormente le extraemos su determinante, obtenemos la ecuación característica y al factorizar obtenemos los valores característicos.

Una vez calculado los eigenvalores, procedemos a calcular los eigenvectores. Se plantea un eigenvector de la siguiente forma (en el caso de una matriz de 2×2).

$$v_1 = \begin{pmatrix} a \\ b \end{pmatrix}$$

Donde los valores a encontrar, son los valores de a y b . El eigenvector cumple la siguiente condición:

$$Av_1 = \lambda v_1$$

La cual nos da la siguiente ecuación matricial:

$$Av_1 - \lambda v_1 = 0$$

Esta ecuación tiene las restricciones que cumplen los valores de \mathbf{a} y \mathbf{b} .

Ejemplo 3

Encontrar los eigenvectores de la matriz:

$$A = \begin{pmatrix} 1 & 4 \\ 3 & 5 \end{pmatrix}$$

Para este caso, ya tenemos calculados los eigenvalores $\lambda_1=-1$ y $\lambda_2=7$. Cada uno de estos eigenvalores, tiene un conjunto de eigenvectores correspondientes. Como se trata de una matriz de 2×2 , el eigenvector propuesto tiene que tener dimensiones de 2×1 :

$$v_1 = \begin{pmatrix} a \\ b \end{pmatrix}$$

Para el primer eigenvalor $\lambda_1=-1$:

$$\begin{pmatrix} 1 & 4 \\ 3 & 5 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = -1 \begin{pmatrix} a \\ b \end{pmatrix}$$

Multiplicando:

$$\begin{pmatrix} a + 4b \\ 3a + 5b \end{pmatrix} = \begin{pmatrix} -a \\ -b \end{pmatrix}$$

De esta forma:

$$\begin{pmatrix} a + 4b \\ 3a + 5b \end{pmatrix} - \begin{pmatrix} -a \\ -b \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} 2a + 4b \\ 3a + 6b \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Y nos da las siguientes ecuaciones:

$$\begin{aligned} 2a + 4b &= 0 \\ 3a + 6b &= 0 \end{aligned}$$

Que nos da $\mathbf{a}=-2\mathbf{b}$, para ambas ecuaciones. Entonces el eigenvector está formado de la siguiente forma:

$$v_1 = \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} -2b \\ b \end{pmatrix} = b \begin{pmatrix} -2 \\ 1 \end{pmatrix}.$$

En forma general el valor de b puede ser cualquier valor real. Así entonces, un eigenvector correspondiente a $\lambda_1=-1$, es:

$$v_1 = \begin{pmatrix} -2 \\ 1 \end{pmatrix}.$$

Para el segundo eigenvalor $\lambda_2=7$:

$$\begin{pmatrix} 1 & 4 \\ 3 & 5 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = 7 \begin{pmatrix} a \\ b \end{pmatrix}$$

Multiplicando:

$$\begin{pmatrix} a + 4b \\ 3a + 5b \end{pmatrix} = \begin{pmatrix} 7a \\ 7b \end{pmatrix}$$

De esta forma:

$$\begin{pmatrix} a + 4b \\ 3a + 5b \end{pmatrix} - \begin{pmatrix} 7a \\ 7b \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} -6a + 4b \\ 3a - 2b \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Y nos da las siguientes ecuaciones:

$$\begin{aligned} -6a + 4b &= 0 \\ 3a - 2b &= 0 \end{aligned}$$

Que nos da $a = \frac{2}{3}b$, para ambas ecuaciones. Entonces el eigenvector está formado de la siguiente forma:

$$v_2 = \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \frac{2}{3}b \\ b \end{pmatrix} = b \begin{pmatrix} 2/3 \\ 1 \end{pmatrix}.$$

En forma general el valor de b puede ser cualquier valor real. Así entonces, un eigenvector correspondiente a $\lambda_2=7$, es:

$$v_2 = \begin{pmatrix} 2/3 \\ 1 \end{pmatrix}.$$

Otro vector que pertenece a la familia, es cuando $b=3$.

$$v_2 = \begin{pmatrix} 2 \\ 3 \end{pmatrix}.$$

Cualquiera de estos dos eigenvectores son los correspondientes a $\lambda_2=7$.